

Visual LANSA Framework Guide

- [Introduction](#)
- [What's New](#)
- [If You Want Your Project to Succeed](#)
- [Getting Started](#)
- [Development Architecture](#)
- [Key Concepts](#)
- [Building the Application](#)
- [Tutorials](#)
- [Frequently Asked Questions](#)
- [Framework Programming](#)
- [Advanced Topics](#)
- [Troubleshooting](#)
- [Application Performance](#)
- [Definitions](#)
- [Appendix](#)

Please send your comments and suggestions to LANSA Support at:
lansasupport@lansa.com.au.

Disclaimer: While every effort has been made to ensure that the information in this material is accurate, in no event shall LANSA be liable for any damages arising from its use. LANSA makes no warranties, expressed or implied.

Edition EPC132100

Edition Date November 6, 2014

© LANSA

Introduction

What is the Visual LANSA Framework?

What Does It Look Like?

Should You Use the Framework?

Who Is It For?

What Are Its Benefits?

What is the Visual LANSA Framework?

The Framework is an optional extension to LANSA which provides an application Framework for designers and developers.

You should have a look at the Framework if you are new to LANSA, have just completed a training course and asking yourself this question:

How do I design and implement my first Visual LANSA or LANSA for the Web application?

The Framework helps you to:

Prototype You can use the Framework to prototype commercial applications very rapidly without writing a single line of code. Your executable prototype can be used to communicate your intentions to end-users and to other developers.

Design The Framework has an MS-Outlook style user interface familiar to your end-users. Its application model is simple and easy to understand.

Modernize Optionally you can use the RAMP tools in the Framework to help you rapidly modernize any existing System i applications you might have. No changes to your 5250 applications are required to do this.

Implement and deploy The Framework minimizes the amount of coding you will have to do. As each part of your prototype application is approved you simply snap it out of the Framework and snap in a real component that you have coded. The Program Coding Assistant can be used to automatically generate most of the code.

Maintain and enhance The structured environment makes maintaining and enhancing the application efficient and controlled.

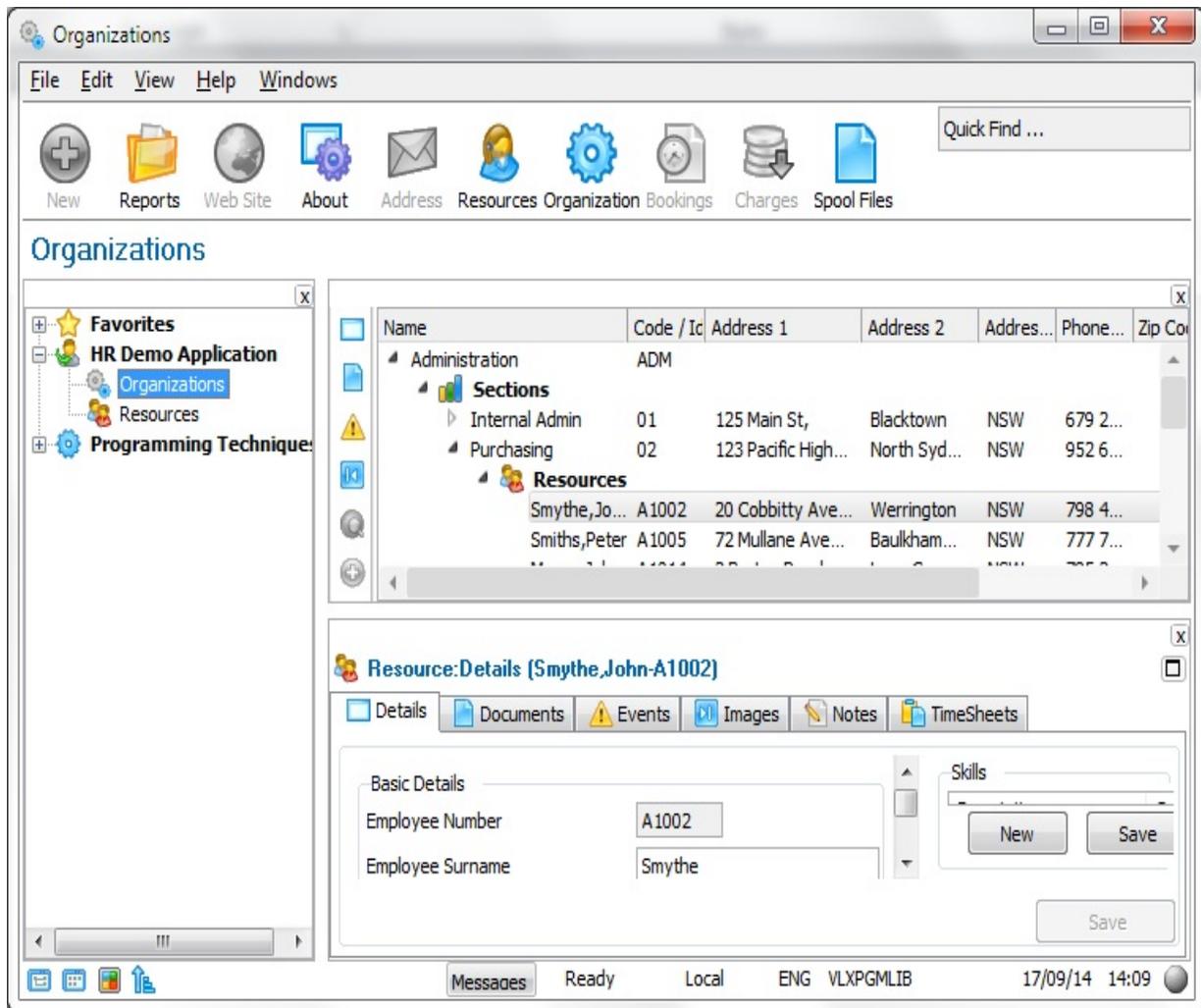
A key benefit of the Framework is that from a single application model it supports deployment to:

1. Windows Rich (or Smart) Client – which is referred to as VLF.WIN in this guide.
2. Web Browser – referred to as VLF.WEB
3. Microsoft .NET - referred to as VLF.NET

A Framework can be deployed to any or all of these environments, in any combination.

What Does It Look Like?

This is what Framework in Windows looks like:



The Web versions of the Framework look almost identical to the Windows one.

Should You Use the Framework?

Consider using the Framework to:

- Develop commercially focused, medium to large integrated applications
- Minimize learning curve and maximize early productivity
- Enable significant end-user involvement during design
- Rapidly deliver a solution
- Use a single application model for both Windows and Web

The Framework is an optional extension to the Visual LANSA and LANSA for the Web products. It is not the optimal solution for all types of applications because it is focused on providing rapid application development for common commercial applications.

It hides and abstracts some of Visual LANSA and LANSA for the Web's flexibility to achieve higher levels of productivity and an improved learning curve.

By using the prototyping facilities in the Framework you should be able to quickly assess if it is suitable for your type of application. If it is not, then simply use Visual LANSA and/or LANSA for the Web in the traditional manner.

The Framework should not be used:

- For specialized system tools or features.
- For small or stand alone non-integrated applications.
- For sites that already have their own application Framework or equivalent.
- For developers already very experienced in using Visual LANSA or LANSA for the Web.
- When building brochure style, B2C or publicly accessible web sites.

See also [Should you use Windows or Web Browser Applications?](#)

Who Is It For?

[Starting the Framework](#)

Designer

You can use the Framework to quickly prototype an application. No programming knowledge is required.

Developer

The developer turns the prototype into a real application by creating components to be snapped in the Framework.

Typically you would use the Framework if you:

- Possibly have some experience with basic event-driven programming, for example VB
- Have completed Visual LANSA or LANSA for the Web training course
- Are thinking "What do I do now?"

Several developers can work simultaneously to create the modular snap-in components.

Administrator

Some of the users of the deployed Framework can be given authority to administer user access and the servers defined for the applications. Defining servers and controlling user profiles and passwords is optional.

End-user

The end-user of the deployed Framework can only see the applications you have created. The design time environment, such as the Framework menu, the Assistant and the Tutorials are hidden.

What Are Its Benefits?

Web and Windows applications from a single application model

The Framework can provide you with a single and consistent application model for both Windows and Web. The advent of WAMs makes the single application model capabilities even stronger than before.

Standard interface

A design loosely based on Microsoft Outlook. Outlook is very popular around the world and almost all users are familiar with it, whether at work or at home.

This model provides a cockpit or dashboard style design where everything that an end-user might need to do is just a few clicks away.

XML-based external design schema

The Framework is instantly executable.

Because of the modular design, many developers can work on different parts of the application at the same time.

The versions of the prototype can be quickly emailed for evaluation and feedback.

Rapid prototyping

Applications, business objects and commands can be defined in a few minutes and can be used in emulation mode before any code to support them actually exists.

A vision of how the completed result will look, act and feel can be formed and executed before a single line of code is written.

This process also acts as a way of rapidly uncovering new or hidden business requirements.

Prototype becomes the application

You do not have to discard any part of your prototype. When you are ready to turn the prototype into a real application, you simply snap your custom-made parts in the Framework. This means you keep the basic structure of the application, its business objects, commands, menus and images.

Rapid modernization	<p>You can use the Framework RAMP tools to quickly enable your System i applications for Windows.</p> <p>Absolutely no change to the 5250 application is required and yet RAMP offers advanced navigation, search and organization capabilities that go well beyond other modernization tools.</p>
Simple to code	<p>The Framework gives the developer much easier access to advanced Visual LANSA and LANSA for the Web features. For example, it implicitly handles multi-form and multi-component interactions and referencing.</p>
Load-on-demand architecture	<p>A load-on-demand architecture that dynamically loads application components as they are used. This prevents having to load your entire application during start up.</p>
Standards for development and user interfaces	<p>The modular structure of the snap-in filters and command handlers encourages standardized development practices.</p> <p>The user interface is to a large extent controlled by the Framework which enforces consistency.</p>
Productivity improvements in addition to Visual LANSA	<p>The Framework handles all the basic functions of the application, such as multi-form interactions and referencing.</p>
A huge "jump start" for new Visual LANSA or LANSA for the Web	<p>The environment helps the developers in getting started with the application development and guides them towards a standard implementation.</p>

developers

Gradual and
benefits driven
introduction to
some of the
heavier OO
concepts

The Framework is based on OO concepts such as inheritance. This underlying structure and its benefits become gradually more obvious to the Framework developers as they progress in implementing the application.

What's New

This section outlines new features in the EPC132100 version of the Framework. Also see [Features No Longer Supported](#).

Unicode support

The Framework can now handle Unicode data for certain parameters in certain methods. Unicode data can also be used in code tables, the Virtual Clipboard and in tracing. See [Using Unicode Data with the Framework](#).

Tailor User/Authority reports

It is now possible to tailor the User/Authority reports produced by the Framework. The output can be a .csv file or it can be written out to a database. The report structure and content is fully customizable. See [Customized User/Authority Reporting](#).

IBM i password management

New options allow IBM i passwords to be managed from the Framework logon screen. IBM i user profile error checking (eg: wrong password, password expired) has also been improved. See [IBM i Password Management](#).

Reusable parts as code table data handlers

Reusable parts can now be used as code table handlers. This simplifies coding because most default processing is done by the ancestor component. See [Code Table Data Handlers Can Now Be LANSAs Reusable Parts](#).

Overriding object captions

You can provide additional information to the end-user by [Temporarily Overriding Object Captions](#).

The Programming Code Assistants feature has been updated

All WEBEVENT code assistants have been removed. All images and screen shots have been updated to match the latest release of LANSAs.

The graphic quality of all images and screen shots has been improved.

Indicate development status of objects

Developers can now attach development status indicators and notes to Framework objects. The indicators and notes are visible when the Framework is run in development mode.

See [Development Status](#).

Improved security

VLF-WEB/NET temporary state files are now encrypted.

New theme

The new 2014 clean theme is a low key, clean, crisp and flat theme that is designed to reflect the style used in later Windows products. The predominant color is white.

See [New Theme 2014 Clean](#).

Visual Styles can be changed at run time

If you need a visual style to change at run time, you can swap in a new style or styles from any command handler, filter, or snap in instance list.

See [Change a visual style at run](#)

Updated organizations business object instance list

The shipped Organizations Business Object Instance List provides improved summary and preview information when you hover over an item in the tree.

See [Updated Organizations Business Object Instance List](#)

uQueryCanDeactivate reason codes

You can now find out what the user is trying to do when the uQueryCanDeactivate check occurs using the optional [Reason Code](#) parameter.

Instance list with Direct-X features

The shipped Resources business object's snap in instance list browser now uses (and requires) Direct-X features.

See [Shipped Resources Instance List Browser Updated](#).

Improved demo images

The demo images have been improved.

time.

Option to enable WEBEVENT functions

The ability to use WEBEVENT filters or command handler is a now a deprecated feature.

If you have an existing Framework that already contains WEBEVENT filters or command handlers and you wish to add more, you can [Enable Framework for WEBEVENT Functions](#) on the on the Framework Properties tab.

Alternative find path algorithm in RAMP-TS

Developers of large RAMP-TS systems using Axes 3.1 can now use an alternative Find Path algorithm, Find Path V2.

The new algorithm is much faster, but may occasionally result in RAMP finding a different path between screens than it did with the old algorithm.

[So thorough re-testing of the RAMP application is required if changing from one algorithm to the other.](#)

If users in large RAMP systems are experiencing delays the first time they go from one screen to another, it is possible that the number of RAMP nodes in the session are making it difficult for the Find Path algorithm to find a path between the screens. Using the new version of the algorithm may solve this problem.

To activate Find Path V2 use this statement in your login script:

```
GLOBAL_bUseFindPathV2 = true;
```

In the application trace, the line "Using Find Path V1" or "Using Find Path V2" will indicate which algorithm is being used.

Reusable parts as instance list relationship handlers

Instance list relationship handlers can now be created as reusable parts.

The code for the reusable part version is simpler, and similar to the code used by filters to write to the instance list.

See [Reusable Parts as Instance List Relationship Handlers](#).

Features No Longer Supported

The support of these features has been removed in EPC132100:

WEBEVENT functions deprecated

The ability to use WEBEVENT filters or command handler is a now a deprecated feature.

If you have an existing Framework that already contains WEBEVENT filters or command handlers and you wish to add more, you can [Enable Framework for WEBEVENT Functions](#) on the Framework Properties tab.

Auto Session Signoff

Over time the auto session signoff feature has been rendered ineffective by pop-up blocking and browsers that do not allow secondary windows to be opened.

Now when a VLF-WEB session is closed, an attempt is made to delete any associated temporary files it has in the temporary folder. If the close is initiated from a 'signoff' or 'exit' VLF command, the temporary files should be removed successfully. If the session close is initiated by closing the browser window, then whether the temporary files are deleted depends on the web browser being used: some browsers do not allow further requests to be sent from a closing window.

To counter any build-up of temporary files, use the shipped VLF or OS features to regularly clear unwanted files from the temporary folder. For example see the FAQ [Can I purge old information from my temporary directory in a batch job?](#)

Web Load Image

The web load image feature has been removed.

WAMTRANS=C

The previously deprecated WAMTRANS=C (Client Side XML transformation) option has been removed from VLF-WEB and VLF.NET.

Form Layout Assistant

The previously deprecated FLA (Form Layout Assistant) feature has been

Rich Text RAD-PADs

Rich Text RAD-PADs are no longer supported.

removed from VLF-
WEB and VLF.NET.

GZIP File Compression

GZIP File Compression
has been removed.

VLF-WEB – All old Windows XP style themes have been removed

The old XP theme set which was restricted to using
Internet Explorer 9 (or earlier) have been removed.

All browsers are now handled the same way with the
same options.

This is reflected in this old developer VLF-WEB
launch form:



The screenshot shows a 'Choose Browser' dialog box with the following options:

- Chrome
- Firefox
- Safari
- Internet Explorer
 - Windows XP Look
 - Blue
 - Olive Green
 - Silver
- Web Look

being simplified to this in V132100:



The screenshot shows a simplified 'Choose Browser' dialog box with the following options:

- Chrome
- Firefox
- Safari
- Internet Explorer

All associated XXXXXXXX_WEB.HTM suffixed
start up and launch files are no longer generated and
can no longer be used.

IBM i Password Management

The Framework has now the ability to change IBM i passwords and to check their expiry date.

The User Administration Settings tab in Framework properties has these new settings:

IBM i User Profile Management

Allow IBM i password change

Change Password IIP

Extended validation

Check Password Expiry

Warn before (days)

Allow IBM password change	Check this box to allow password change.
Check Password Expiry	Check this box to compare the password's expiry date with the current date.
Warn before (days)	Specify how many days before the expiry date to start issuing warnings.

New IBM i Host Server Mapper properties

In the Server Details tab you need to specify the IBM i server to connect to for password expiry checks and/or password change requests:

IBM i Host Server Mapper

Name / IP address

Port

Name / IP address	Name or IP address of the IBM i Server Mapper. Supports full 40 character long IPV6 type addresses.
Port	Port of the IBM i Server Mapper to connect to. Defaults to IBM

default.

See:

[IBM i Password Expiry Checking](#)

[Changing the IBM i Password](#)

IBM i Password Expiry Checking

To enable password expiry checking, you need to:

- Select Check Password Expiry in the User Administration Settings tab of the Framework properties.
- Specify a number of days greater than zero to start receiving warnings.
- Specify an IBM i Host Server Mapper name or IP address and the correct port.

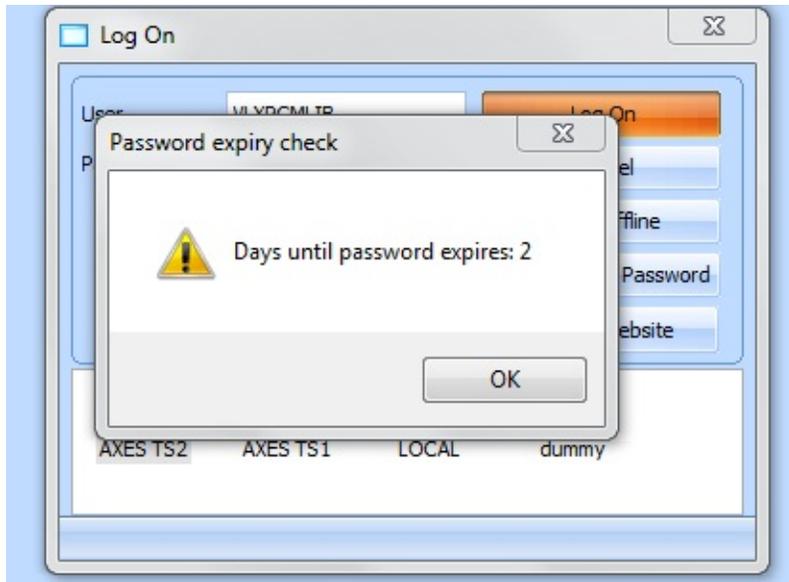
[IBM i Password Expiry Checking in Windows](#)

[IBM i Password Expiry Checking in Web](#)

IBM i Password Expiry Checking in Windows

During log on, when IBM i password checking is enabled, the Framework checks the expiry date of the password of the user logging on and compares it with the current date. If the difference is less or equal than the specified value in Warn before (days) a warning is issued.

For example:

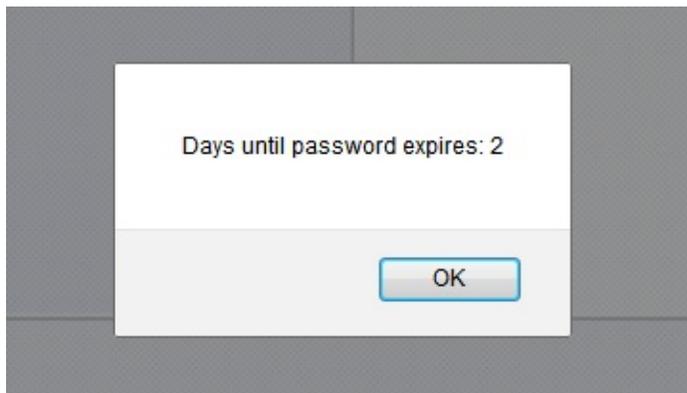


IBM i Password Expiry Checking in Web

In the web platform this feature has been added to the web IIP. Refer to the shipped function UFU0001 in process UF_SYSBR.

Note that fields have been added to the Exchange command.

During log on, the password's expiry date is compared with the current date. If the difference is less or equal than the specified value for Warn before (days), a warning is issued. For example:



Changing the IBM i Password

To enable IBM i password change you must check the Allow IBM i password change option in the User Administration Settings tab of Framework Properties.

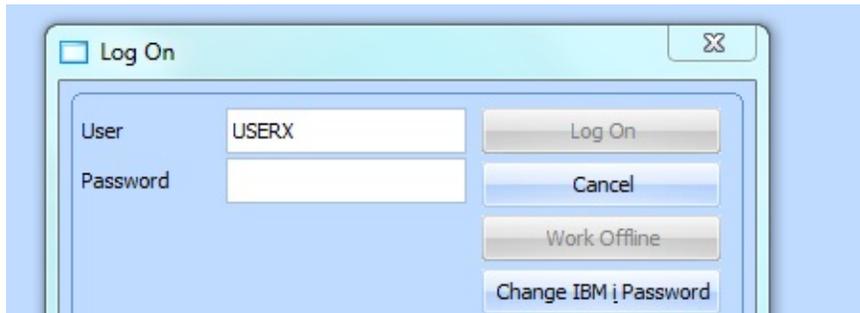


[Changing the IBM i Password in Windows](#)

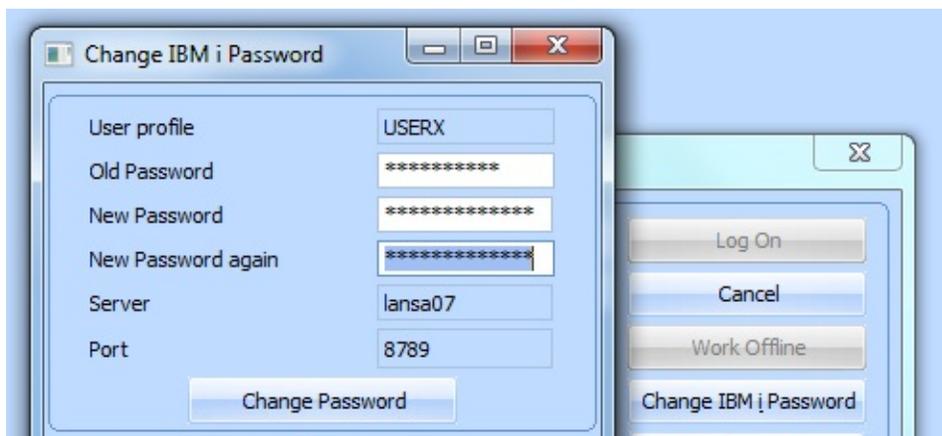
[Changing the IBM i Password in Web](#)

Changing the IBM i Password in Windows

When IBM i password changing is allowed and the IBM i Host Server Mapper name or IP address has been specified, the Change IBM i Password button is displayed on the log on screen:

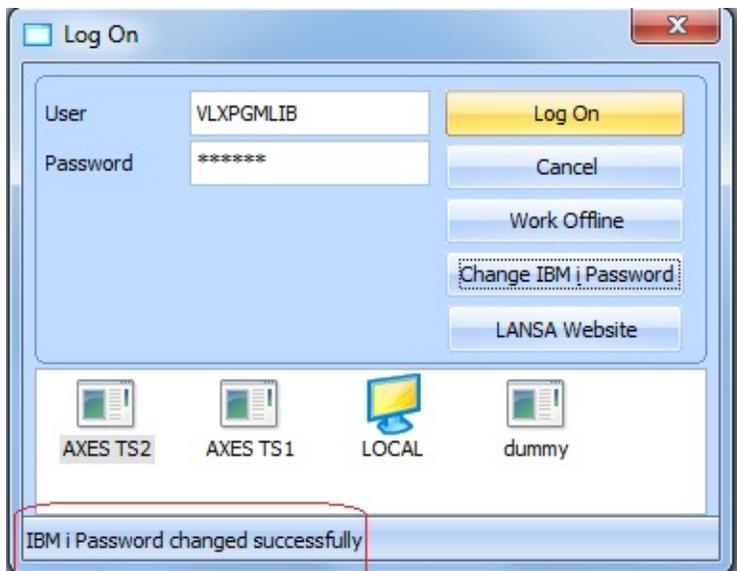


Clicking the Change IBM i Password button brings up the Change IBM i Password dialog:



Type in the old and the new password, and then click the Change Password button.

A message indicates if the change was successful:

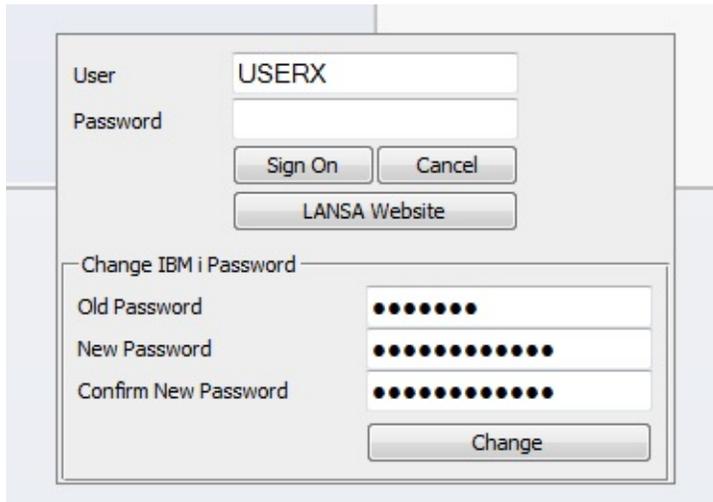


Or unsuccessful:



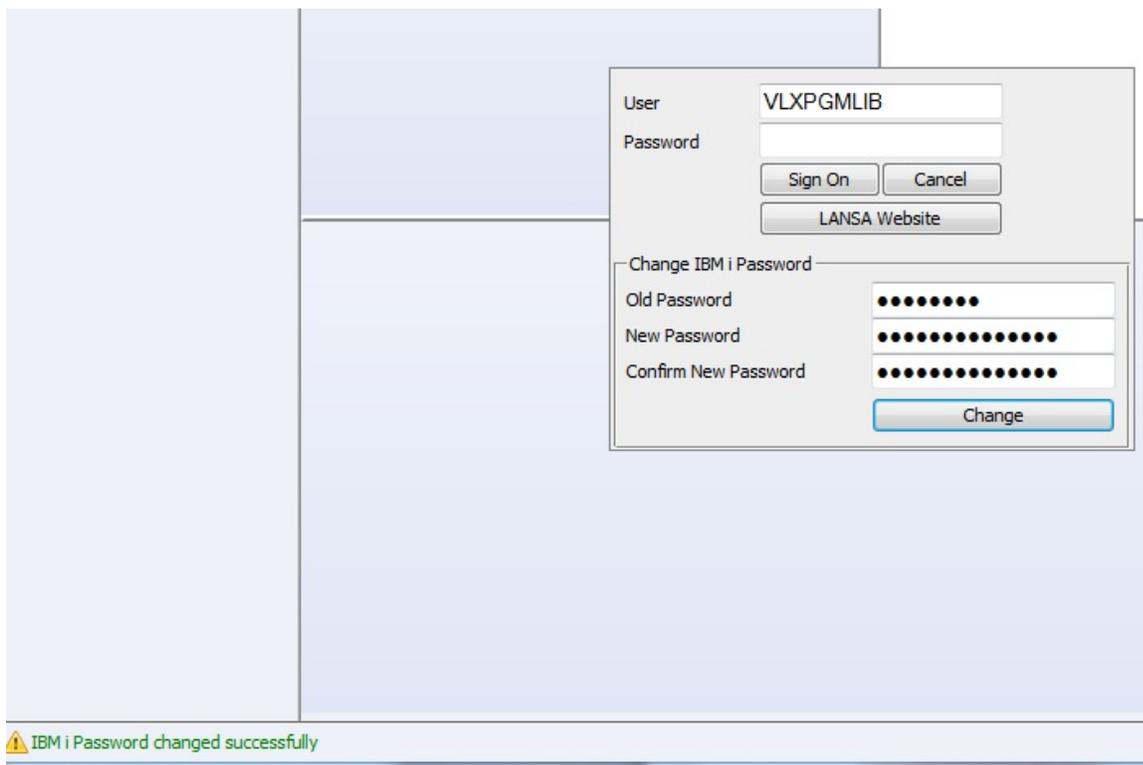
Changing the IBM i Password in Web

On the web, if the server to connect to was saved as the Deployment Server and the IBM i Host Server Mapper name or IP address has been specified, the Change IBM i Password button is displayed in the log on panel:



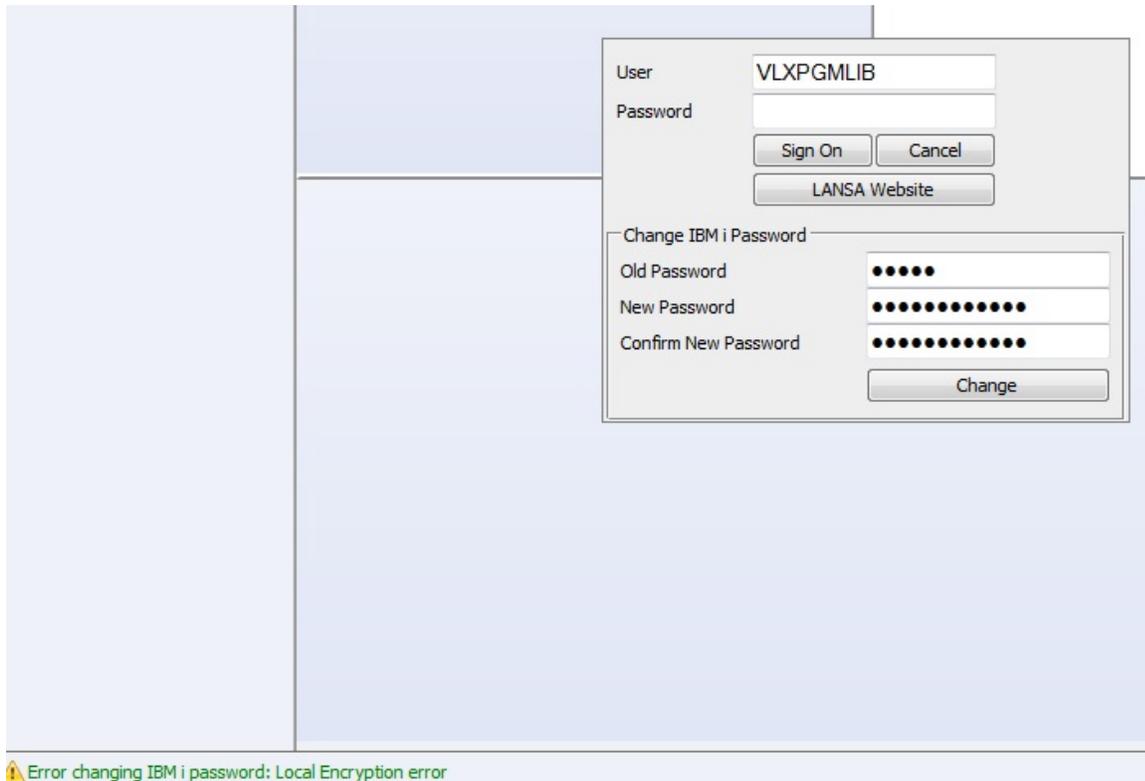
The screenshot shows a dialog box with two sections. The top section is for logging on, with a 'User' field containing 'USERX', an empty 'Password' field, and buttons for 'Sign On', 'Cancel', and 'LANSA Website'. The bottom section is titled 'Change IBM i Password' and contains three password fields: 'Old Password' (8 dots), 'New Password' (12 dots), and 'Confirm New Password' (12 dots). A 'Change' button is located at the bottom of this section.

Type in the old and the new password and press the Change button. A message indicates if the change was successful:



The screenshot shows the same dialog box as above, but now the 'Change' button is highlighted in blue. At the bottom of the window, a green message bar displays a warning icon and the text 'IBM i Password changed successfully'.

Or unsuccessful:



To change the IBM i password on the web, the user must specify the IBM i server and port in the start up URL:

```
+IBMI_SERVER=<ibmiServer>+IBMIPORT=<ibmiPort>
```

Where:

<ibmiServer> is the ip of the server where the profile's password is to be changed

<ibmiPort> is the port for changing IBM i profile passwords (usually 449)

For example:



```
+IBMISERVER=10.2.0.181+IBMIPORT=449
```

See [Web Application Start Options](#) for more details.

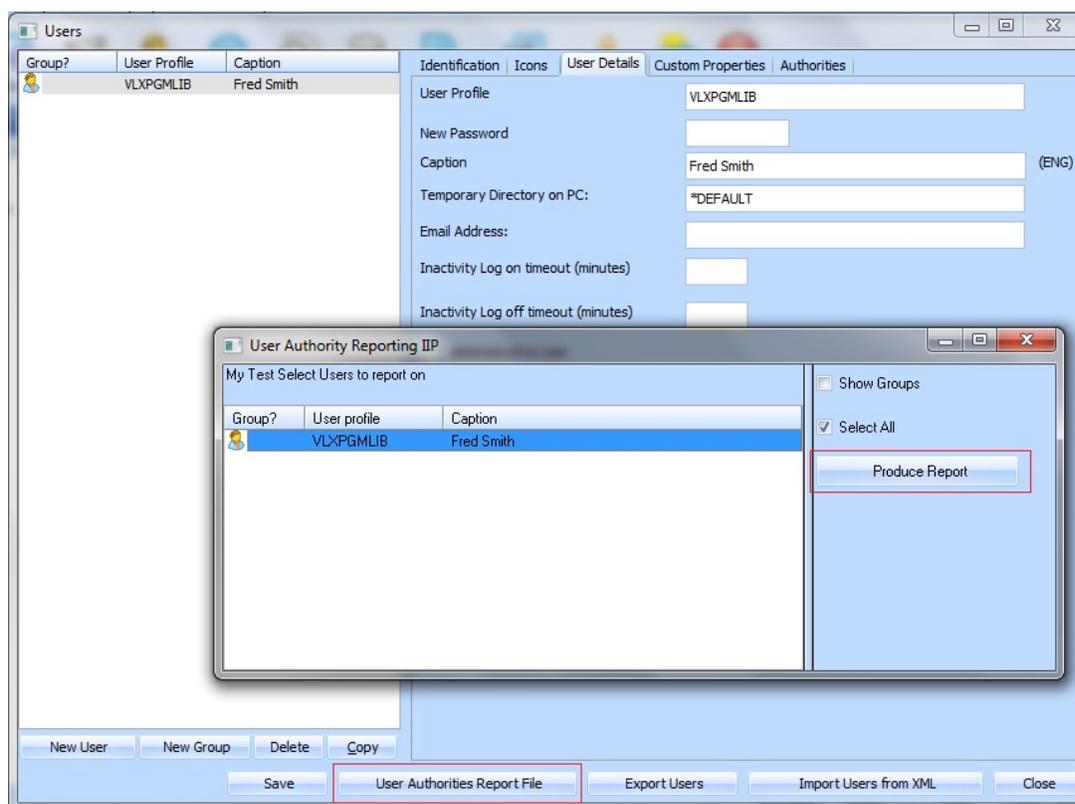
Customized User/Authority Reporting

This feature is designed for advanced developers.

Developers can write their own user/authority report programs for administrators. See [Writing your own version of the User Authority report](#).

The program used for user/authority reporting is specified in the Framework properties --> User Administration Settings --> Authority Settings --> [Report on Users - Imbedded Interface Point \(Id\)](#).

If unspecified, the standard user/authority report is produced.



Development Status

Developers can attach a development status indicator and notes to Framework objects. These are visible when the Framework is run in development mode. For more information see [Development Status Feature](#).

The screenshot shows a window titled "Business Object Properties - The essential business object". The window has several tabs: Identification, Icons, Visual Styles, Filters, Filter Settings, Commands Enabled, Command Display, and Custom. The "Identification" tab is active. The properties are as follows:

Caption	The essential business object	(ENG)
Caption (Singular)	The essential business object	
Hint:	A very simple filter and a very simple command handler	
Sequence:	1	
Internal Identifier:	9A475843B9D5452D9A4926BDE8E5C927	
Unique Identifier:	12	
User Object Name / Type	9A475843B9D5452D9A4926BDE8E5C927	

Below the text fields are several checkboxes:

- Restricted Access
- Allow on Web
- Allow in Windows
- Allow Selection from Navigation Pane

There is a field for "Allow this Object to be Opened in a New Window" with the value "Manually".

The "Last Changed" field contains the value "20140903-145808-VLXPGMLIB".

A red box highlights a development status menu. The menu items are:

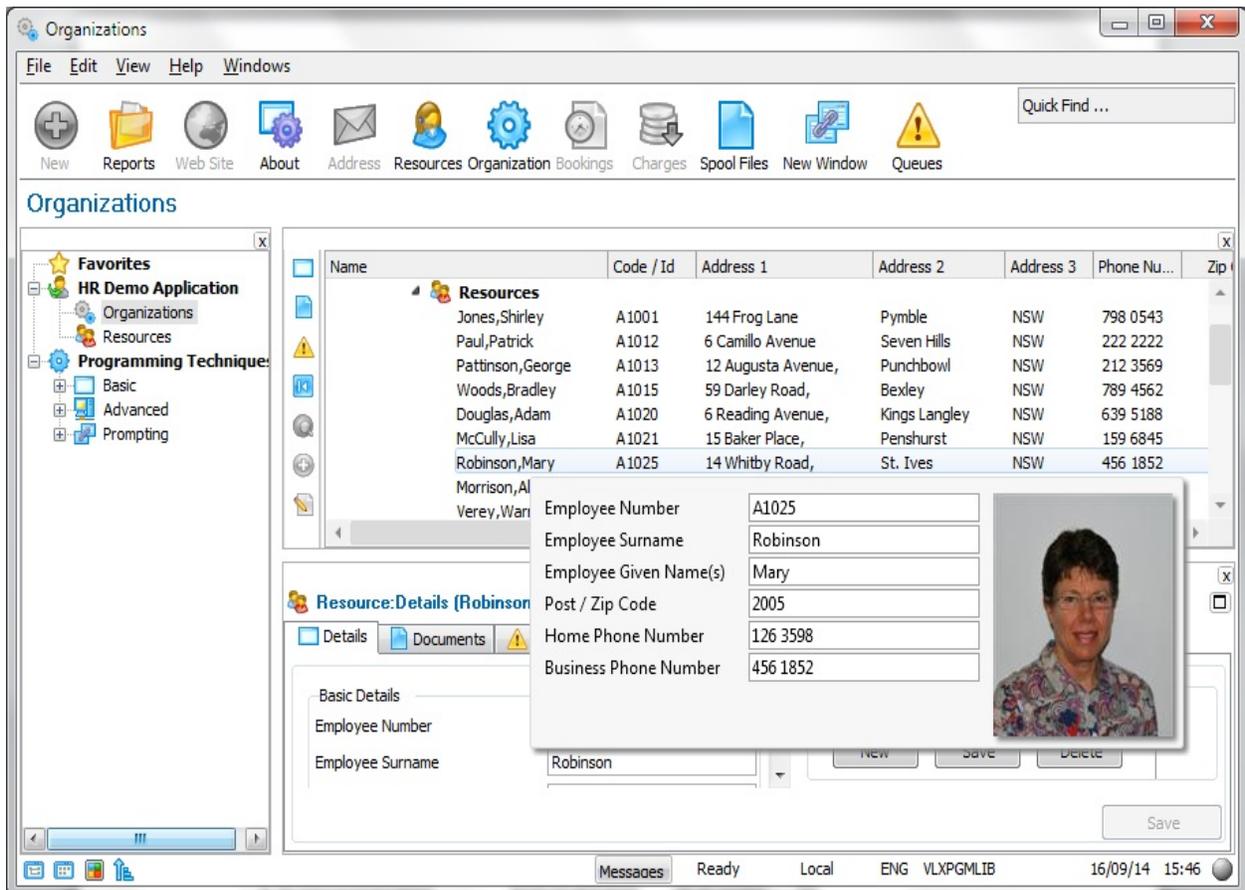
- Closed
- Under Implementation
- Under Test
- Alter Development Status (dropdown arrow)

Below the menu is a text area containing the following notes:

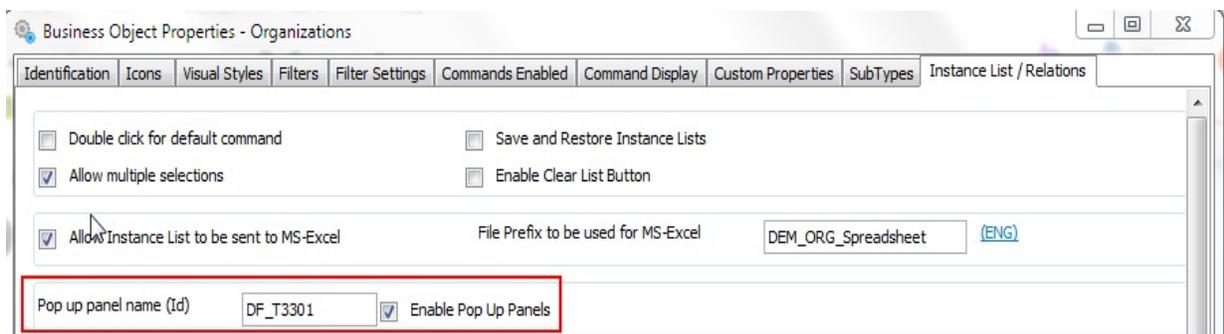
These are the notes for the essential business object.
This is another note.

Updated Organizations Business Object Instance List

The shipped Organizations Business Object Instance List provides improved summary and preview information when you hover over an item in the tree.

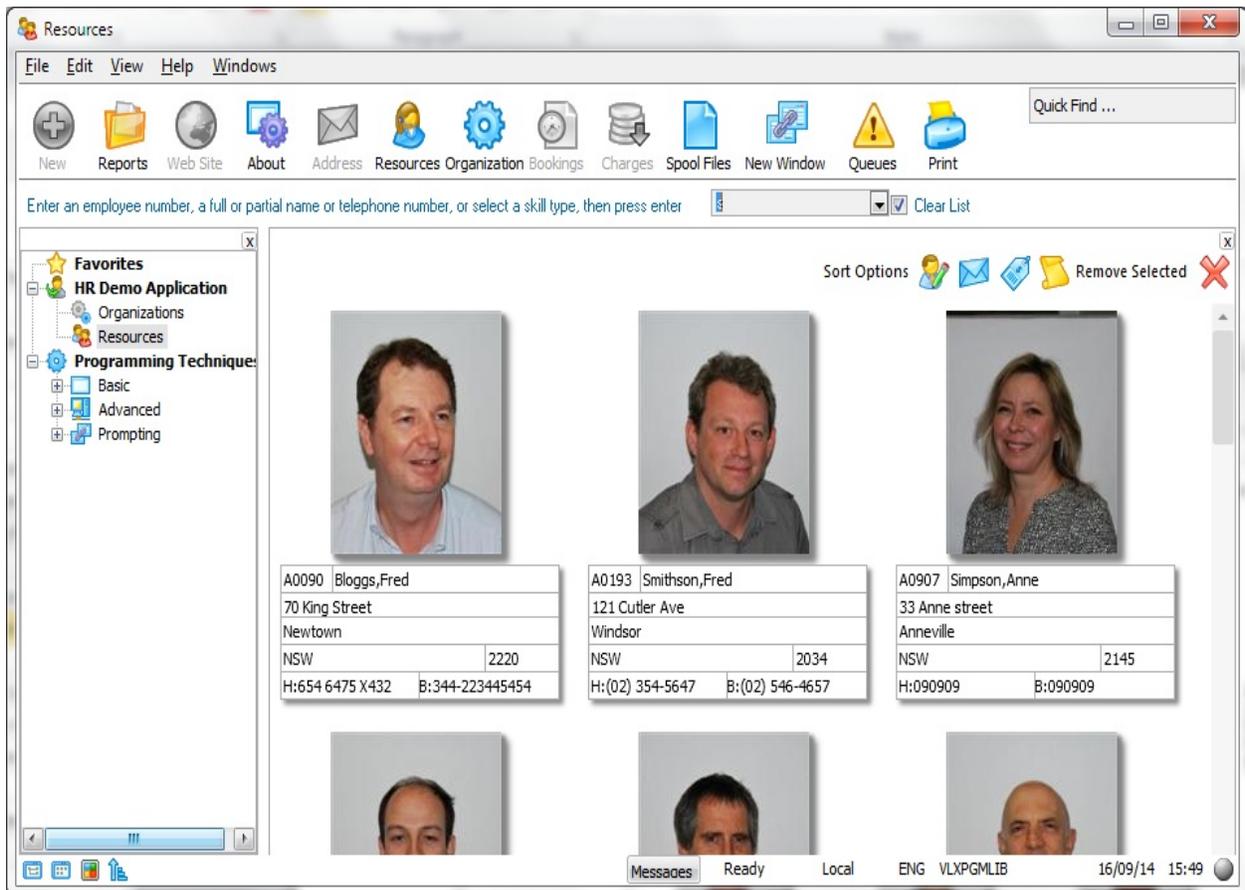


It does this by using the Pop up panel name option on the Instance List / Relations tab:

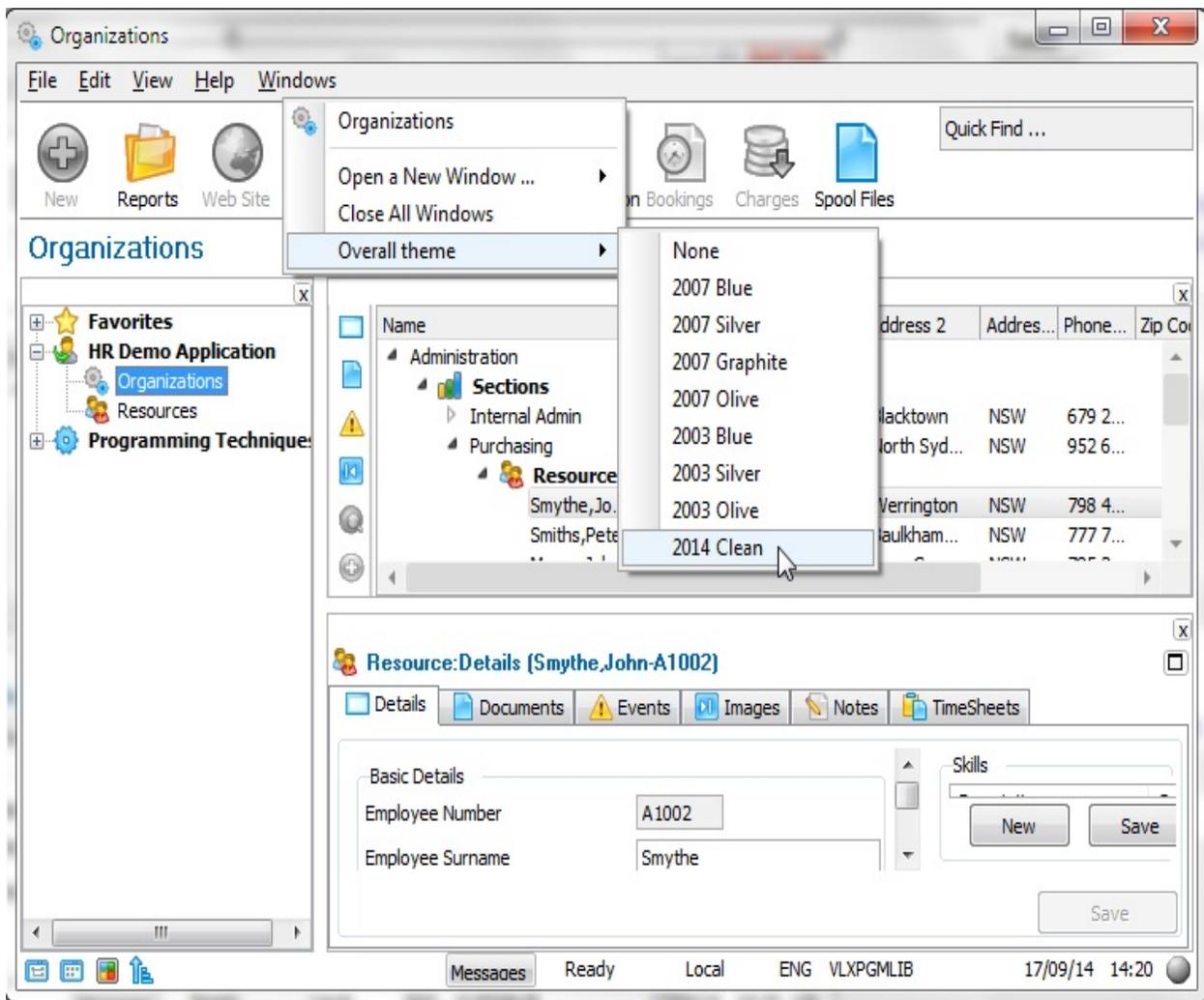


Shipped Resources Instance List Browser Updated

The shipped Resources business object's snap in instance list browser now uses (and requires) Direct-X features.



New Theme 2014 Clean

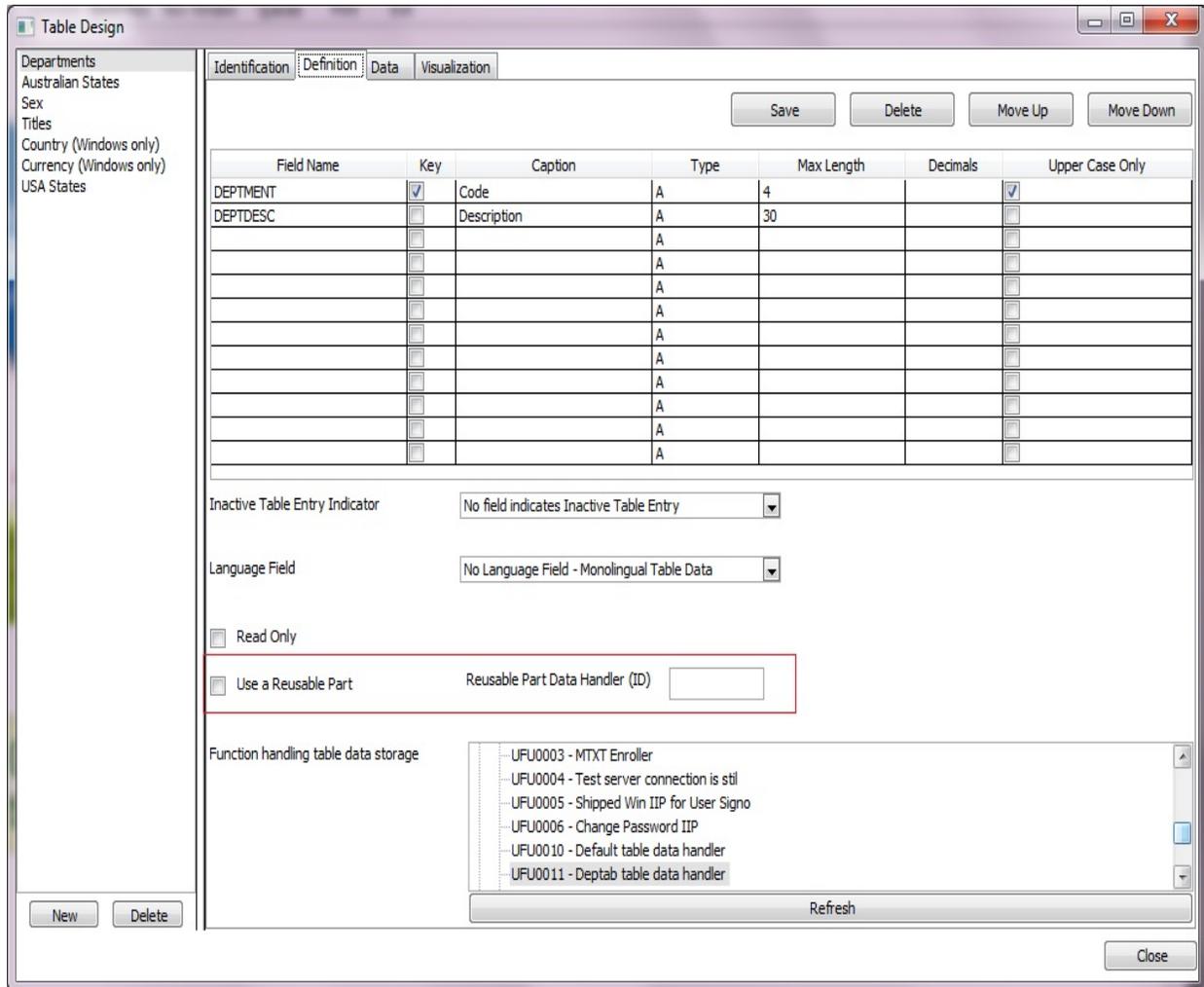


The 2014 clean theme is a low key, clean, crisp and flat theme that is designed to reflect the style used in later Windows products. The predominant color is white and the predominant style is low key and flat.

Classical Windows features like bright colors, highlighting and 3D borders and color gradients that are used for screen ornamentation have largely been removed.

Code Table Data Handlers Can Now Be LANSA Reusable Parts

Using reusable parts as code handlers simplifies coding because most default processing is done in the ancestor component.



Reusable part data handlers can also deal with Unicode data.

To use a Reusable part as a Code Table Data Handler tick the Use Reusable Part property check box and enter the identifier of the data handler component in the Reusable Part Data Handler (ID) property field.

UF_TDH01 is the default Reusable part Code Table Data handler and can be used as is, or as a starting point for a custom data handler.

See [Creating Your Own Table Data Handler Reusable Part](#).

Reusable Parts as Instance List Relationship Handlers

Instance list relationship handlers can now be created as reusable parts.

The code for the reusable part version is simpler, and similar to the code used by filters to write to the instance list.

For example, this is the code used to write the sections for an organization into the instance list, in the shipped demo:

```
WHEN VALUE_IS('= DEM_ORG')
* Expand Sections in a Department/Organization

#DEPARTMENT := #AKEY1
SELECT FIELDS(*ALL) FROM_FILE(SECTAB)
WITH_KEY(#DEPARTMENT)

Signal uAddListItem AKey1(#DEPARTMENT) AKey2(#SECTION)
VisualId1(#SECDESC) VisualId2(#SECTION) AColumn1( #SECADDR1)
AColumn2(#SECADDR2) AColumn3(#SECADDR3)
AColumn4(#SECPHBUS) NColumn1(#SECPCODE)
BusinessObjectType(#TargetType)

ENDSELECT
```

See [Sample Relationship Reusable Part](#).

The VLF looks for a component for the relationship handler (rather than a function) if the following property is checked:

Business object properties --> Instance List tab --> Relationship properties --> Use a Reusable part.

What Was New in Previous Versions

These sections outline new features in earlier versions of the Framework:

[New Features in EPC 870 version of the Framework](#)

[New Features in EPC 868 version of the Framework](#)

[New features in EPC 839 version of the Framework](#)

[New features in EPC 831 version of the Framework](#)

[New features in EPC 826 version of the Framework](#)

[New features in EPC 804 version of the Framework](#)

[New features in EPC 793 version of the Framework](#)

[New features in EPC 785 version of the Framework](#)

New Features in EPC 130100 version of the Framework

This section outlines new features in the EPC130100 version of the Framework.

Version 13 Features

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

Important! See [Using your Visual LANSAs Framework in Direct-X mode](#)

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.

See [Quick Find Override Feature](#).

Launch Applications from the Status Bar

Applications can now be launched directly from the Framework status bar when the Framework is executed using RenderType M.

See [Launching Applications from](#)

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 can disable the feature by right mouse clicking on the instance list, if popups are not required.

See [Popup Panel Hints in the Instance List](#).

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.

[the Status Bar.](#)

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.

New Features in EPC 870 version of the Framework

This section outlines new features in the EPC870 version of the Framework.

Framework on iPads and Android Touch Devices

Framework web applications can be run on iPads and Android touch devices. To enable touch friendly functionality required in touch devices, use the web startup URL parameter TOUCH= to start your Framework application. See [Touch Device Considerations](#).

Web Configuration Assistant

The [Web Configuration Assistant](#) helps you configure the Framework for your web servers when you are starting VLF web development for the first time.

The web Configuration Assistant can be accessed from either the (Framework) menu, or from the (Framework) --> (Framework Properties) --> Framework Details tab.

It can be used instead of the Framework developer preferences and VLF Administrator Console, if preferred.

VLF-WEB Enhancements - Stronger WAM Support

You can now create mini filters using WAMs. See [RDMLX for a WAM Mini Filter](#).

WAMs can handle Web Application help. See [Help Text for Web Applications](#) and the shipped WAM UF_SY0002 for a basic example.

Weblets that use the jQuery UI visual design themes can now be used in VLF WAM Filters and Command Handlers. Simply use vlf_layout_v2 as the WAM's layout Weblet and

User Authorities Report File

VLF-WIN administrators can now produce a .csv (comma separated variable) file which contains a complete list of all the users on the system and their authority to every object on the system. This file can be viewed in MS Excel.

To produce the report, use the User Authorities Report File button on the VLF-WIN

apply a jQuery Theme to the WAM. See LANSAs Web Guide for more details about Theming WAMs. Other style changes may be experienced when using vlf_layout_v2.

(Administration) --> (Users) screen.

New Features in EPC 868 version of the Framework

This section outlines new features in the EPC868 version of the Framework.

Note that:

- You need to recompile/redeploy all relationship handlers after you install this EPC.**
- You must be using a LANSA V12 supported platform, see http://www.lansa.com.au/downloads/support/version12_supportedplatforms.pdf**
- If you plan to run VLF.Web with Chrome, Firefox or Safari, test that your existing hand-coded JavaScript or HTML in WAMs and RAMP scripts work in these browsers.**

Cross-Browser Support for VLF.Web

VLF.Web Runs in Internet Explorer, Firefox, Chrome and Safari.

Note that VLF.NET only supports Internet Explorer.

aXes-TS2 Can Be Used as the RAMP-TS Engine

aXes-TS2 Can Now Be Used in RAMP-TS, enabling the use of aXes-TS2 compatible browsers for RAMP-TS applications.

Instance Lists

You can now add [Date and Date/Time Columns](#) to the standard shipped instance list.

[Edit Codes](#) can be applied to numeric columns.

You can [Control Row Color in Instance Lists](#).

[Selection of Multiple Items Can Now Be Controlled in Instance Lists](#).

[Instance List Toolbar Buttons Can Have Associated Text Descriptions](#) which improves appearance and usability.

VLF.WIN [End-users Can Choose](#)

Web Content on Signon Screens

All users must log on, so signon screens are an ideal place to advertise company business or application changes and to promote communication. You can do this by adding a web page or a button that launches a URL to the signon screen.

[A Button on Framework Signon Screen Can Launch A Web Page](#) in its own window. The button can be added to the signon screen in VLF.WIN, VLF.Web and VLF.NET.

[Web Page Can Be Shown on the VLF.WIN User Logon Screen](#) instead of an image.

[Instance List Columns to Be Displayed](#) allowing them even more control over how their applications appear.

In VLF.WIN [Code Can Directly Access Visual Lansa Trees](#) that are used to visualize the instance list content.

New Value-Add Features Shown in Demonstration Application

The Framework demonstration system shows uses for [IBM i Server Message Queues](#).

The Demonstration Application now includes a new command, [Print Screen](#), which prints out the current VLF screen/window.

The shipped Framework also includes a new command, [Open in New Window](#), which opens the current business object in a new window.

VLF.NET

The generated VLF.NET applications can be compiled to run in a 32-bit platform.

Each business object can have its own screen layout.

Default application text strings (such as User Name and Password labels in the Sign In dialog box) can now be dynamically overridden.

Connecting to the Server

You can use Windows user profiles and passwords when connecting to the server.

For IBM i servers, you can validate the password entered by the user exactly as typed.

The Framework can now handle long passwords also for Windows applications.

You can control which servers appear in the Connect dialog.

See [Connecting to Servers](#).

Commands

You can now find out the current state of a command handler panel (Normal or Maximized) in relation to the Framework.

Commands can bypass Framework or RAMP locks in VLF.WIN applications.

See [Commands and Command Handlers](#).

See [VLF.NET Enhancements](#).

User-Interface

The Framework tool bar style can be set to show large buttons with text descriptions (for new users).

There is an option to hide the Windows Control Bar which is displayed on Framework forms when more than one Framework window is open.

The VLF.WIN application start up logic now positions the initial main window so that is clearly visible to the user even if it has previously been moved to a place difficult to access.

See [User-Interface Enhancements](#)

Other

Visual LANSA Active-X Controls VF_AX003 and DF_XMLAC have been changed to use MSXML6 interface.

The Framework design XML can be saved in an encrypted form.

You can now set a confirmation message when end-users attempt to close the Framework.

See [Other Enhancements](#).

Note that some older features have been marked as deprecated. This means they are supported for backward compatibility, but you should not use them for new development.

VLF.Web Runs in Internet Explorer, Firefox, Chrome and Safari

Framework Web applications now run in Internet Explorer, Firefox, Chrome and Safari.

See [Execute Framework as a Web Application](#).

Also see [Using VLF-WEB Applications with Safari, Firefox or Chrome](#).

If you are planning to use RAMP-TS in other browsers than IE, see [Configuring Web RAMP-TS for Chrome, Safari and Firefox](#).

The screenshot shows a web browser window with the following content:

- Browser Title:** EPC868 Framework
- Address Bar:** /Images/VLF_EX1/VF_SY001_SYSTEM_MPD_LS_A_ENG_BASE.HTM?Developer=Y
- Menu:** File, Edit, View, Actions, Tools, Help
- Toolbar:** New, Open, Stop, Refresh, Home, Back, Forward, Print, Mail, Print, Exit
- Page Title:** Organizations
- Left Navigation Panel:** On Tool Bar, HR Demo Application, Organizations (selected), Resources, Programming Techniques
- Main Table:**

Name	Code / Id	Address 1	Address 2	Address 3	Phone Number	Zip
ADMINISTRATOR DEPT	ADM					
Sections						
INTERNAL ADMIN SRV	01	125 Main St,	Blacktown	NSW	679 2536	
PURCHASING SECTION	02	123 Pacific Highway,	North Sydney, 2000	NSW	952 6475	
Resources						
SMYTHE,JOHN	A1002	20 Cobbitty Avenue,	WERRINGTON,	NSW,	798 4381	
SMITHS,PETER	A1005	72 Mullane Avenue,	BAULKHAM HILLS,	NSW,	777 7265	
MOORE,JOHN	A1014	2 Burton Road,	LANE COVE,	NSW,	785 2695	
- Resource Details (SMYTHE,JOHN - A1002):**
 - Details:** Employee Number: A1002, Employee Surname: SMYTHE, Employee Given Name(s): JOHN, Employee Salary: 25,000.04, Start Date: 1977/01/01
 - Events:** (Warning icon)
 - Images:** (Image icon)
 - Notes:** (Note icon)
 - TimeSheets:** (TimeSheet icon)
 - Basic Details:** (Section header)
 - Skills:**

Skill	Grade	Met requirement	Expiry Date
Administratrn Part 1	Distinction	Met requirement	1998/03/2
Administratrn Part 2	Fail		1998/05/0
Computer Science Deg	Pass		1998/05/0
English Degree	Distinction		1998/05/0
Company Introduction	Pass	Met requirement	1998/02/0
Keyboard Skills	Pass	Met requirement	1998/02/0
Management Course 1	Distinction	Met requirement	1998/03/1
- Address and Contacts:** (Section header)
- Status Bar:** Messages, Ready, 23rd February 111 | 8:44

Using VLF-WEB Applications with Safari, Firefox or Chrome

This version of the VLF-WEB supports IE, Safari, Firefox and Chrome as web browsers.

There are some things you need to understand about delivering your VLF-WEB applications for use in multiple browsers:

If you already have a VLF-WEB application running under IE

You will have created LANSa WAM components as filters and command handlers and may have used RAMP scripting. It is likely you have also embedded your own hand-coded JavaScript and/or HTML inside this VLF-WEB application. You need to test that your hand-coding is multi-browser capable and may need to alter it.

Note that:

- The skin known as WIN has been deprecated. WEB and XP (Blue, Olive and Silver) are still available.
- At the time of the release of EPC868, IE version 9 is still at Candidate Release level. **Tests of EPC868 were conducted using version 8.** It is strongly recommended that you choose version 8 until version 9 has been officially released and tested. Although version 7 is supported, we will not action any support requests for version 7 unless they can be replicated in version 8.
- When you save the framework in EPC868 there are only two .HTM files generated:

<framework name>_<language>_BASE.htm – to execute with IE using the XP skin, Firefox, Chrome and Safari.

<framework name>_<language>_WEB.htm – to execute with IE using the WEB only.

Safari is the primary browser of the Apple Mac, iPad and iPhone

Safari is the browser used by Apple Macs, iPhones and iPads. These are the reference platforms for Safari.

If you report an issue with Safari you can only reasonably expect us to action it if it is reproducible on a reference platform.

You have to test on multiple browsers

You should not deploy to a browser without having tested your application on it.

Please note that some browser features used by the VLF may have problems. When such issues are found we will try to find a work-around for it, but we may decide there is too much effort for little reward. For example, there is an ongoing problem with modal windows in Chrome which would require too much effort to address.

You have to test on the real platform

While you might conduct the majority of testing for a touch device (say) using an emulator you should not deploy an application without testing it on a real touch device.

You have to learn to debug multiple browsers and multiple platforms

Web browsers are all behaviourally different. A probable consequence of testing your application on multiple browsers is that you will need to learn how to trace and debug your applications on each browser.

Supporting multiple browsers comes at a cost

All of the preceding points indicate that supporting multiple browsers comes at a cost.

You should not support a browser unless there is a justifiable business case and you have tested against it.

Configuring Web RAMP-TS for Chrome, Safari and Firefox

To use RAMP-TS it is necessary to bypass browser cross-domain security (security relating to documents accessing documents from a different domain).

Domain refers to the Host:Port combination. For example if the VLF uses a host MyHost in port 81, the VLF domain is MyHost:81 and if RAMP-TS (aXes) also uses MyHost but in port 8080, the RAMP-TS domain is MyHost:8080.

Therefore the VLF and RAMP-TS are accessing different domains.

In Internet Explorer cross-domain security is bypassed by adding the host to Trusted sites:



This cannot be done in Chrome, Firefox or Safari, so it is necessary to use the web server Reverse Proxy feature to bypass cross-domain security. The Reverse Proxy settings for the sample host names look like this in the IBM i Admin instance:

Proxy ?

SSL Proxy SSL Proxy Advanced FRCA Reverse Proxy Cache

General Settings Forward Proxy Reverse Proxy Proxy Chaining

Reverse proxy capabilities: Enabled ?

Proxy requests to remote servers: ?

	Request type	Local virtual path	Remote server URL
Example	Client requests	/mirror/foo	http://www.myserver.com/
Example	Redirected requests	/mirror/foo	http://www.myserver.com/
<input type="radio"/>	Client requests	/ts/	http://MyHost:8080/ts/
<input type="radio"/>	Client requests	/agi/	http://MyHost:8080/agi/
<input type="radio"/>	Client requests	/axests/	http://MyHost:8080/axests/

Add

Proxy action for Via headers: Preserve existing Via header lines ?

Outgoing connection buffer size: 0 Bytes or... ?

Override remote error information: Disabled ?

Preserve host headers: Disabled ?

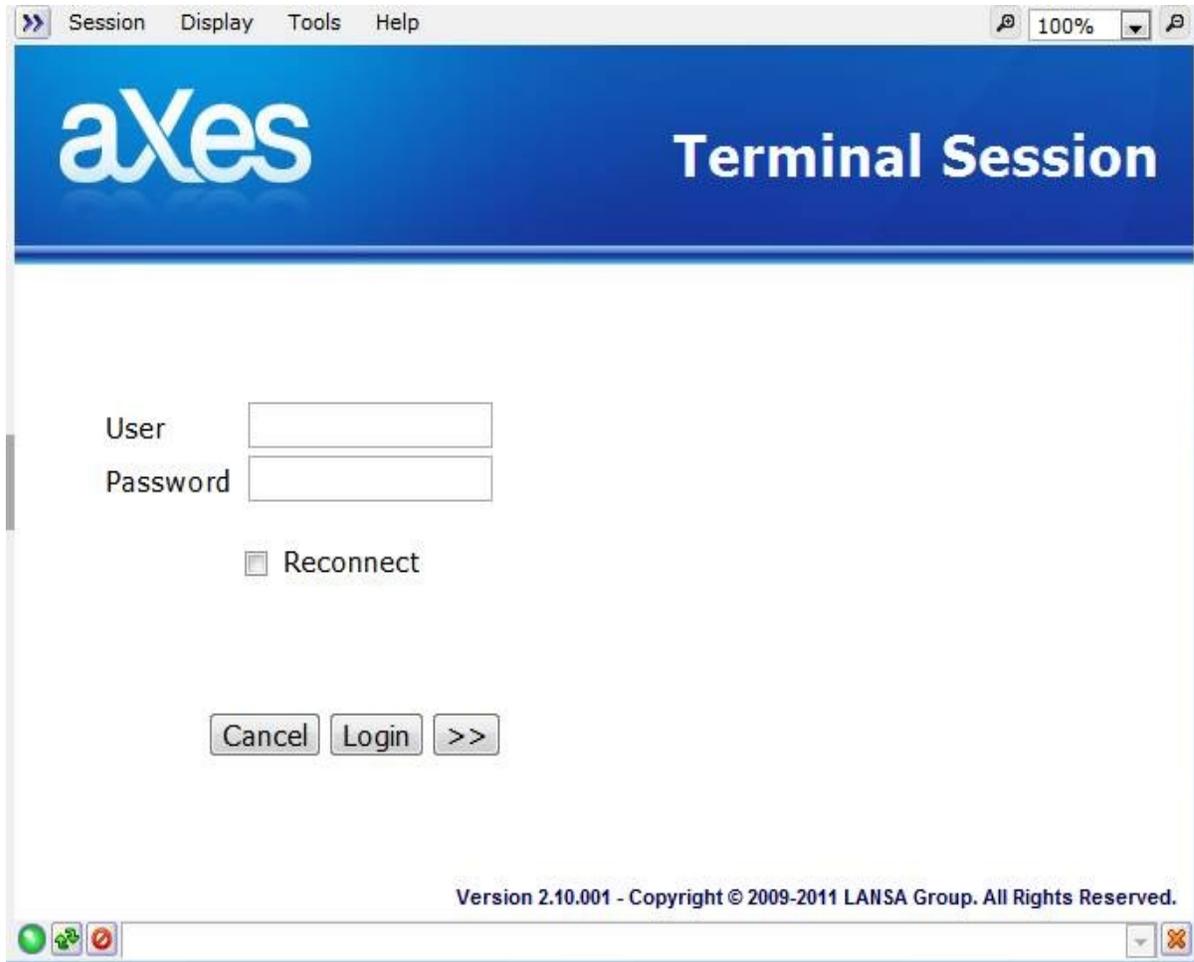
Remote server timeout: 30 Seconds ?

To set up your reverse proxy, replace MyHost:8080 with your host details. Note that the order in which the entries are specified is relevant, and if you specify more than one host, RAMP-TS will attempt just the first one it matches.

Once you've set up the Reverse Proxy you should:

1. Restart the web server
2. Clear the browser's cache
3. Start Fiddler!
4. Try first serving the equivalent of MyHost:8080/ts/ts2/index.html and then MyHost:81/ts/ts2/index.html.

MyHost:8080 is the RAMP-TS (aXes) domain therefore MyHost:8080/ts/ts2/index.html should work straight away and you should see a page like this:

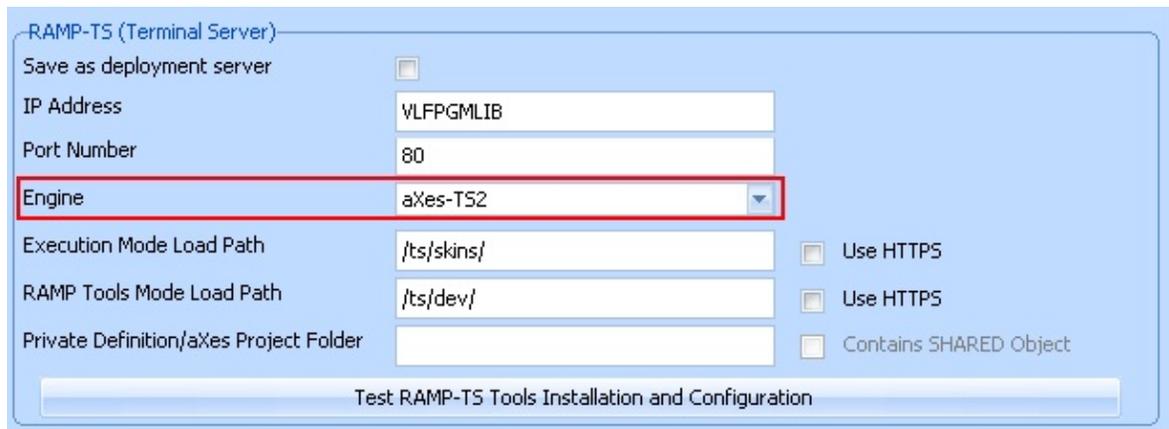


If you cannot see this screen, you may not have aXes installed or there is a problem with your configuration.

MyHost:81 is the VLF domain. If you can serve the same page using the VLF domain it means the change to the web server configuration is working.

aXes-TS2 Can Now Be Used in RAMP-TS

aXes-TS2 Can Be Used as the RAMP-TS Engine, enabling the use of aXes_TS2 compatible browsers for RAMP-TS applications.



The image shows a configuration window titled "RAMP-TS (Terminal Server)". It contains several fields and checkboxes:

- Save as deployment server:**
- IP Address:** VLFPGMLIB
- Port Number:** 80
- Engine:** aXes-TS2 (highlighted with a red box)
- Execution Mode Load Path:** /ts/skins/ Use HTTPS
- RAMP Tools Mode Load Path:** /ts/dev/ Use HTTPS
- Private Definition/aXes Project Folder:** Contains SHARED Object

At the bottom, there is a button labeled "Test RAMP-TS Tools Installation and Configuration".

VLF.WEB applications may use aXes-TS2 (aka RAMP-TS2) with the IE, Safari, Chrome or Firefox web browsers.

VLF.WIN and VLF.NET applications using aXes-TS2 (aka RAMP-TS2) use IE only - they are based on Microsoft web interface technology.

See RAMP-TS property [Engine](#).

Date and Date/Time Columns

Note that all existing relationship handlers need to be recompiled and redeployed because of this enhancement.

Designers can now add Date and Date/Time values to the standard shipped instance list as additional columns:

	Code / Id	Address 1	Address 2	Address 3	Phone N...	Zip Code	Start Date
	A0090	70 MAIN STREET	NEWTOWN NSW	AUSTR...	654 64...	2220	3rd August 1992
IES	A0234	test	test	test	34324	2000	2nd February 1990
	A1001	144 Frog	PYMBLE.	NSW.	798 0543	2001	4th February 1988
	A1012	6 Camillo Avenue	SEVEN HILLS.	NSW.	222 2222	2147	1st May 1986
	A1013	12 Augusta Avenue,	PUNCHBOWL.	NSW.	212 3569	2016	1st December 1985
	A1015	59 Darley Road,	BEXLEY.	NSW.	789 4562	2030	12th December 1984
	A1021	15 Baker Place,	PENSHURST.	NSW.	159 6845	2153	1st March 1980
	A1025	14 Whitby Road,	ST IVES.	NSW.	456 1852	2005	1st May 1986
	A1027	47 Lincoln Street,	STANMORE.	NSW.	489 2485	2007	1st February 1987
	A1111	1 Main Rd	Hill Top	NSW	957 3188	2345	25th September 1989
	A1234	6 Melissa Place	West Pennant	NSW &	(02) 96	2125	14th August 1996

You can specify the display format and, if required, a UTC time conversion:

Sequence	Type	Caption	Width % (Total 100%)	Decimals	Edit Code	Date/Time Output Format	UTC Conversion
50	ACOLUMN3	Address 3	10		Default	SYSFMT8	Local -> Local
60	ACOLUMN4	Phone Number	10		Default	SYSFMT8	Local -> Local
70	NCOLUMN1	Zip Code	10		Default	SYSFMT8	Local -> Local
80	DCOLUMN1	Start Date			Default	DDXXbMMMMMMMMbCCYY	Local -> Local
	ACOLUMN5				Default	SYSFMT8	Local -> Local
	ACOLUMN6				Default	SYSFMT8	Local -> Local
	ACOLUMN7				Default	SYSFMT8	Local -> Local
	ACOLUMN8				Default	SYSFMT8	Local -> Local

The display format and a UTC time conversion are selected from drop-down lists:

Date/Time Output Format	UTC Conversion
SYSFMT8	Local -> Local
MMMMMMMMMMbCCYY	Local -> Local
CCYYDDMM	Local -> Local
CCYYMM	Local -> Local
CCYYMMDD	Local -> Local
CCYYsDDsMM	Local -> Local
CCYYsMMsDD	
DDD	
DDDDDDDD	
DDMMCCYY	

Date/Time Output Format	UTC Conversion
SYSFMT8	Local -> Local
DDXXbMMMMMMMMMMbCCYY	-> Local
SYSFMT8	Local -> Local
SYSFMT8	Local -> UTC
SYSFMT8	UTC -> UTC
SYSFMT8	UTC -> Local

Output formats map to all the formats available for the AsDisplayString

intrinsic function for both RDMLX Date and DateTime fields.

The Date and Date/Time additional columns can be added and updated from RDML and RDMLX filters and command handlers.

There is a required input format for Date and Date/Time values. See [Adding Additional Columns to Instance Lists](#) and [Columns for Instance Lists](#) for full details.

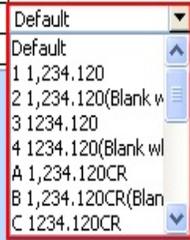
There is also a new subroutine, SETDCOL, available for relationship handler functions. See [Sample Relationship Handler Function](#).

Please read the Required User Action section in the documentation shipped with this EPC.

Edit Code

Edit codes can be applied to numeric additional columns:

Sequence	Type	Caption	Width % (Total 100%)	Decimals	Edit Code	Date/Time Output Format
10	VISUALID1	Name	25		Default	SYSFMT8
20	VISUALID2	Code / Id	10		Default	SYSFMT8
30	ACOLUMN1	Address 1	20		Default	SYSFMT8
40	ACOLUMN2	Address 2	15		Default	SYSFMT8
50	ACOLUMN3	Address 3	10		Default	SYSFMT8
60	ACOLUMN4	Phone Number	10		Default	SYSFMT8
70	NCOLUMN1	Zip Code	10		Default	SYSFMT8
80	DCOLUMN1	Start Date			Default	DDXXbMMMMMMMMbCCYY



See [Columns for Instance Lists](#).

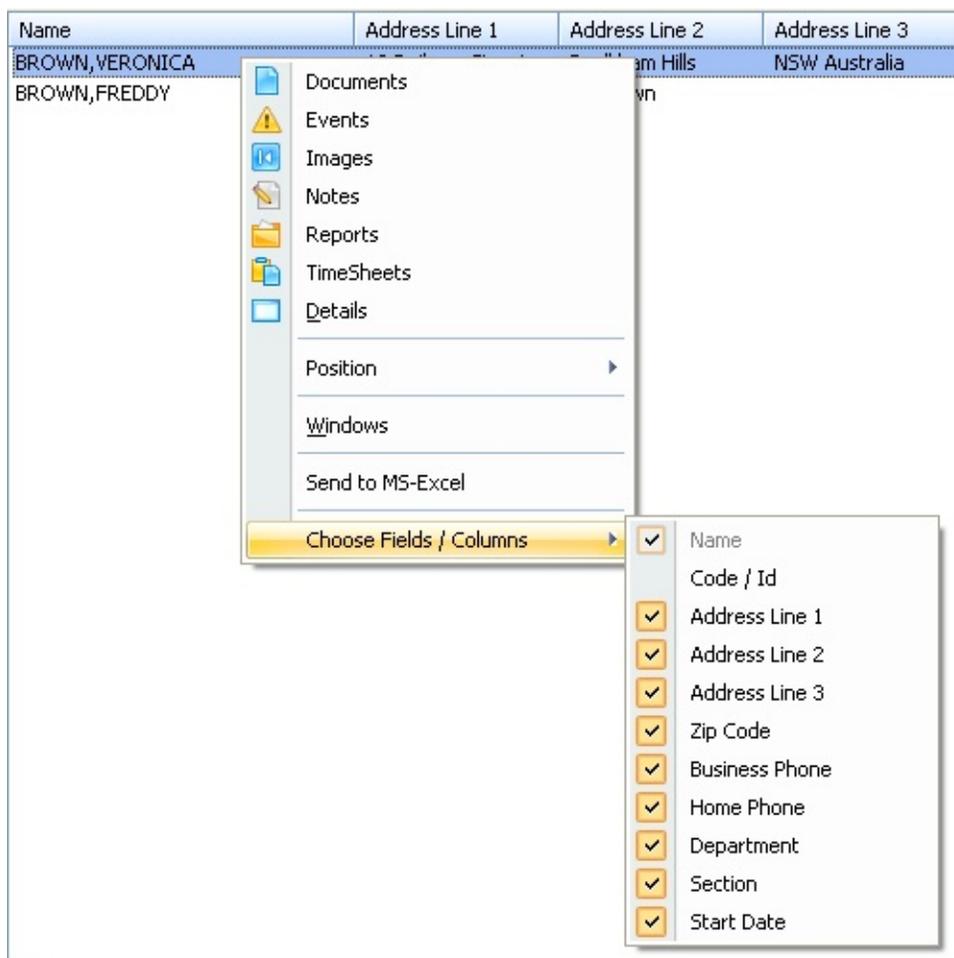
End-users Can Choose Instance List Columns to Be Displayed

VLF.WIN only.

End-users can now choose which columns are to be displayed in instance lists allowing them even more control over how their applications appear. The choice is initiated by the right mouse menu option Choose Fields / Columns.

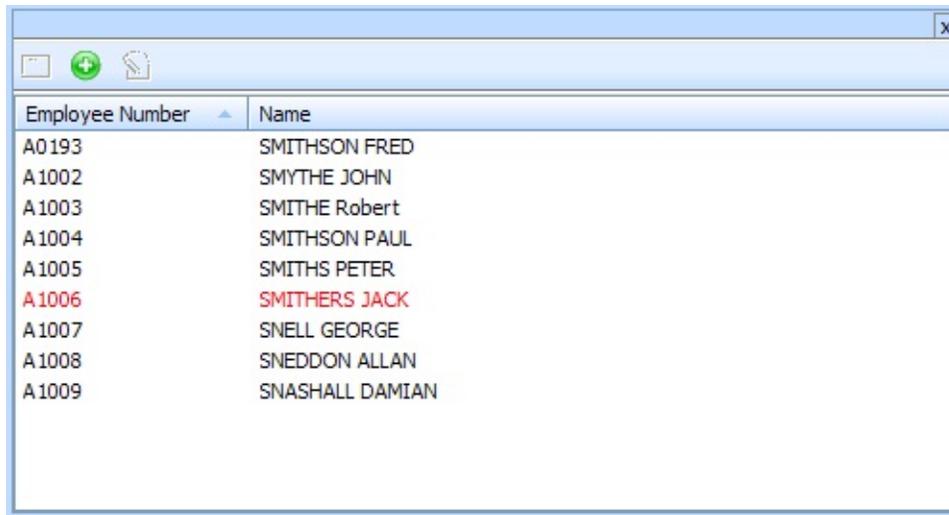
To choose the columns in an over-under or side-by-side child list, position the mouse over the child list before clicking the right mouse button.

The visual identifier 1 column must always be displayed - this stops users from removing all columns from this displayed list.



Control Row Color in Instance Lists

The font and background color of instance list rows can be controlled:



A screenshot of a window displaying a list of employee records. The window has a standard title bar with a close button (X) on the right. Below the title bar is a toolbar with three icons: a folder, a green plus sign, and a document with a pencil. The main area contains a table with two columns: 'Employee Number' and 'Name'. The table has a light blue header. The data rows are as follows:

Employee Number	Name
A0193	SMITHSON FRED
A1002	SMYTHE JOHN
A1003	SMITHE Robert
A1004	SMITHSON PAUL
A1005	SMITHS PETER
A1006	SMITHERS JACK
A1007	SNELL GEORGE
A1008	SNEDDON ALLAN
A1009	SNASHALL DAMIAN

Instance List row colors are specified when adding rows:

- Windows and WAM Filters and command handlers can use a new RowColor parameter on the avListManager.AddtoList and avListManager.UpdateListEntrydata methods.
- Web event filters and command handlers use the instruction VF_SET ROWCOLOR.
- Relationship handler functions call the SETROWCOL subroutine to set the color for WEB and WIN environments.

The color of an instance list entry can also be set back to what it was, by specifying the value "DEFAULT".

For windows environments the color is the name of a visual style that has been enrolled in the VLF. For web environments the color is a CSS string, like:

```
"color:RED;background-color:BLUE;font-style:italic "
```

Note that in the VLF.NET only certain CSS attributes are supported.

You could use row colors to, for example, create heatmaps:

Heat Map

Clear List Search

Employee Surname

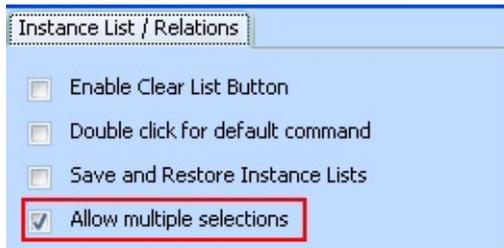
Compare with

Employee Number	Name
A1011	PERRIN CHRISTOPHER
A1010	PERRY WILLIAM
A1509	REDFORD ROBERT
A1023	REID DAVID
A1025	ROBINSON MARY
A1003	SMITHE Robert
A1006	SMITHERS JACK
A1005	SMITHS PETER
A0193	SMITHSON FRED
A1004	SMITHSON PAUL
A1002	SMYTHE JOHN
A1009	SNASHALL DAMIAN
A1008	SNEDDON ALLAN
A1007	SNELL GEORGE
A1024	TAYLOR JOHN
A1022	THOMPSON KELLY
A1016	TURNER JACK
A1020	TURNER VALENTINE

See [Changing the Color of List Entries \(RowColor\)](#).

Selection of Multiple Items Can Now Be Controlled in Instance Lists

Refer to the new property [Allow Multiple Selections](#) on the Instance List/Relations tab of the business object properties window.



Instance List Toolbar Buttons Can Have Associated Text Descriptions

Instance list toolbars may now optionally show the text associated with the toolbar icon – improving appearance and useability.



Name	Address Line 1	Code / Id	Business Phone	Address Line 2
BROWN,VERONICA	12 Railway Street	A0070	(02) 9647 2788	Baulkham Hills
BLOGGS,FRED JOHN ALAN	70 MAIN STREET	A0090	654 6475 X432	NEWTOWN NSW

See [Instance List Tool Bar Text Location](#).

Code Can Directly Access Visual Lansa Trees

In VLF.WIN code you can directly access the VL trees which are used to visualize the instance list content. Once you have a reference to the tree you can access the items within the tree and the columns within the items.

See [Low Level Direct Access to the Visualization Trees](#).

Button on Framework Signon Screen Can Launch A Web Page

Button on Framework signon screen can launch a web page. It is available in all Framework environments.

The launched page could, for example, show a website with the latest company details, or link to an application which is external to the Framework.



See [Launch Button Caption](#), [Launch URL \(Windows\)](#) and [Launch URL \(Web / .Net\)](#):

Web Page Can Be Shown on the VLF.WIN User Logon Screen

On the VLF.WIN User logon screen a web page can be shown instead of an image.



See [Image / Web Page to Display on Form](#)

Since all users must log on, using a web document at this point is an ideal place to advertise company business or application changes and to promote communication.

It also has the benefit of being external to the Framework application - so it is easy to change, even on a daily basis, without requiring any software deployment to the users' PCs.

Using a small web document containing hyperlinks that open secondary browser windows is a good approach. For example:

[Click here for the inside story of our biggest sale this year.](#)

[Click here to read about the latest changes to this application.](#)

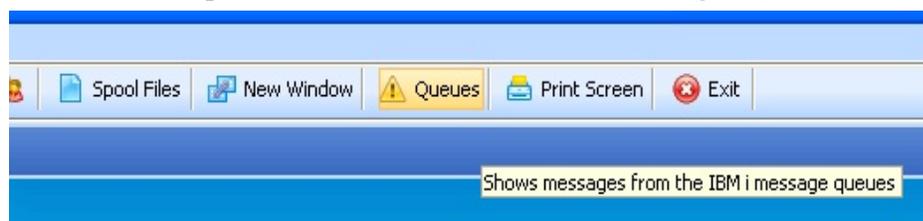
[Click here to review our latest HR policies.](#)

Demonstration System

IBM i Server Message Queues

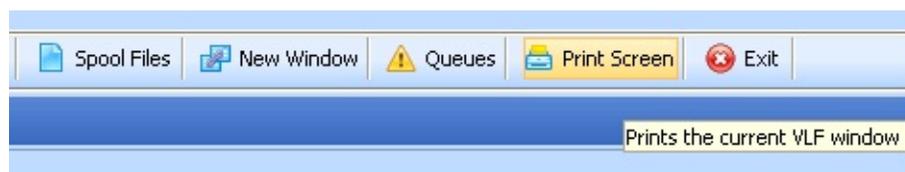
The Framework demonstration system shows some uses for IBM I Server message queues.

The application, which is invoked by the new Queues button on the tool bar, demonstrates the use of personal user queues, group chat queues and event notification queues. See [IBM i Server Message Queues](#).



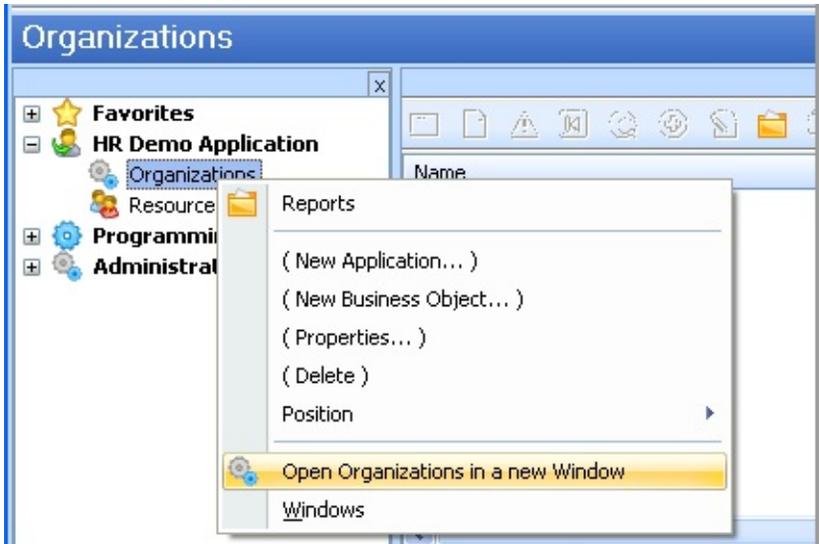
Print Screen

The Demonstration Application now includes a new command, Print Screen, which prints out the current VLF screen/window. This command is accessed via a Toolbar button which is visible when the Demonstration Application is selected.



Open in New Window

The shipped Framework also includes a new command, Open in New Window, which opens the current business object in a new window. If no business object is selected the current application is opened in the new window. If no application is selected, the Framework itself is opened in the new window.



Connecting to Servers

You will find these new properties on the Server Details tab:

Use Windows User Profile and Password

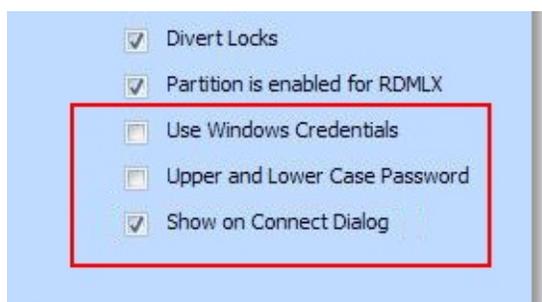
You can use the [Use Windows Credentials](#) option to use a Windows user profile and password when connecting to the server (Kerberos / Single Signon / SSO). The server must have been configured for Single Sign On and the user enrolled first.

Validate Upper and Lower Case Passwords

For IBM i servers, you can use the option [Upper and Lower Case Password](#) to validate the password entered by the user exactly as typed.

Control Which Servers Are Shown in the Connect Dialog

In VLF.WIN the server property [Show on Connect Dialog](#) controls whether a server appears in the list of servers that the end-users see in their Connect dialog.



Long Passwords Are Handled in VLF.WIN

The Framework can now handle long passwords for Framework Windows applications. This means that 32-character passwords can now be handled in all contexts (Windows, RAMP, Web).

VLF.NET Enhancements

Compile for 32-Bit Platform

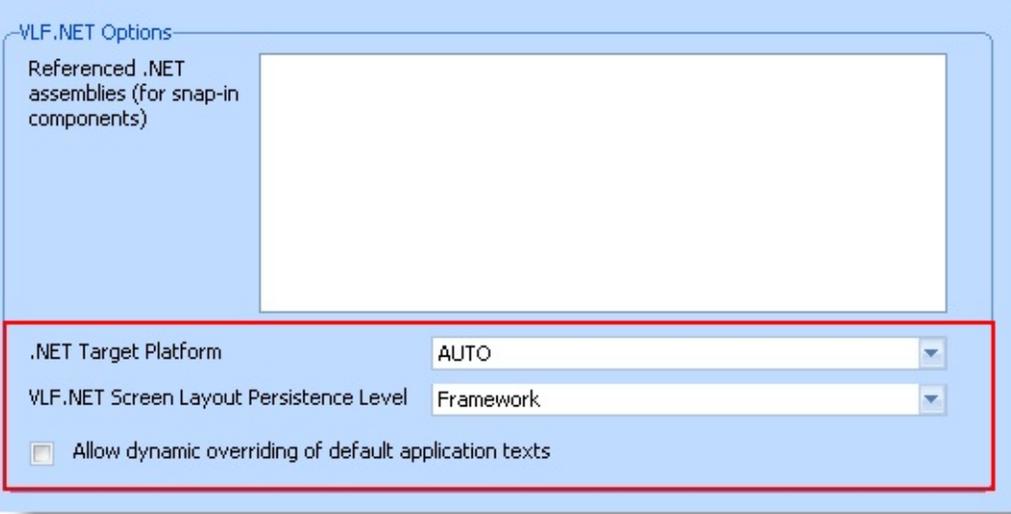
You can now specify target platform for generated VLF.NET applications and force the generated VLF.NET application to run in a 32-bit platform. See [.NET Target Platform](#).

Business Object Can Have Its Own Screen Layout

You can specify that each business object has its own screen layout using the [VLF.NET Screen Layout Persistence Level](#) option. By default a single screen layout applies to the entire Framework.

Default Texts Can Be Overriden

Default application text strings (such as User Name and Password labels in the Sign In dialog box) can now be overridden dynamically by storing the replacement texts in the file VF_MULTI_YYY.js (where YYY is the language code) and placing it on the webserver. See [Allow Dynamic Overriding of Default Application Texts](#).



The image shows a screenshot of the 'VLF.NET Options' dialog box. The dialog has a light blue background and a title bar. It contains several settings:

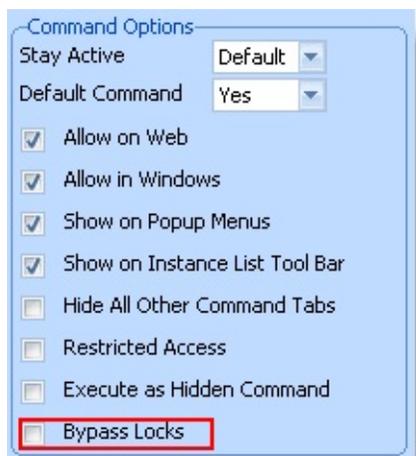
- Referenced .NET assemblies (for snap-in components):** A large empty text area.
- .NET Target Platform:** A dropdown menu currently set to 'AUTO'.
- VLF.NET Screen Layout Persistence Level:** A dropdown menu currently set to 'Framework'.
- Allow dynamic overriding of default application texts:** A checkbox that is currently unchecked.

A red rectangular box highlights the bottom section of the dialog, which includes the two dropdown menus and the checkbox.

Commands and Command Handlers

Bypass Locks

Commands can bypass Framework or RAMP locks in VLF.WIN applications. The [Bypass Locks](#) option can be used to allow certain instances of commands to execute even though the Framework or RAMP may be locked.



The commands Spool Files, New Window, Queues and Print Window on the shipped Framework tool bar are all set to bypass locks and demonstrate the intended use of the property.

Find out if Command Handler Is Maximized

A new Framework Services property, [avCmdPanelState Property](#), will return the current state of a command handler panel in relation to the Framework. When a panel has been maximized, this property will return MAX, and NORM if it has not been maximized.

User-Interface Enhancements

Framework Tool Bar Style

You can set the style for the Framework tool bar to Simple to display large buttons with text:



This style is easy for new users to use. However, the buttons do not wrap on secondary lines so the number you can display is limited by the width of the device.

A text description is shown for all the buttons (derived from the command caption).

If you set the tool bar style to Advanced you can display small buttons which can wrap onto secondary lines:



Text descriptions are only shown for buttons that have a Tool Bar Button Caption defined.

See [Tool Bar Style](#) in the Framework Details tab.

Hide Windows Control Bar

There is now an option to hide the Windows Control Bar which is displayed on Framework forms when more than one Framework window is open. See the options for the Framework property [Multiple Window Control Bar Location](#).

Number of Additional Windows a User can have Open Concurrently: 20

Multiple Window Control Bar Location: None - Do not display control bar

Your Framework Version Number: 1 . 0 . 0 . 73

Automatically Increment when Saving

Show in Help About Text

Last Changed: 20110219-112935-VLFPGLIB

Window Is Positioned at Startup

The VLF.WIN application start up logic now tries to position the initial main window so that is clearly visible to the user.

Previously, a restarted VLF main window may have been positioned so that it was hard to locate and/or use if a user had closed it in a position very close to, or off the edge of the screen, on a secondary monitor that was not connected later, or at a place that was outside the screen after a change to the screen's resolution.

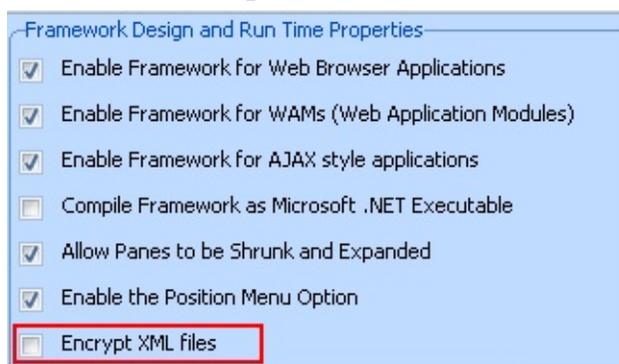
Other Enhancements

MSXML6

Visual LANSA Active-X Controls VF_AX003 and DF_XMLAC have been changed to use MSXML6 interface. Your VL application code probably does not reference these components directly, but if it does it will need to be recompiled and re-tested.

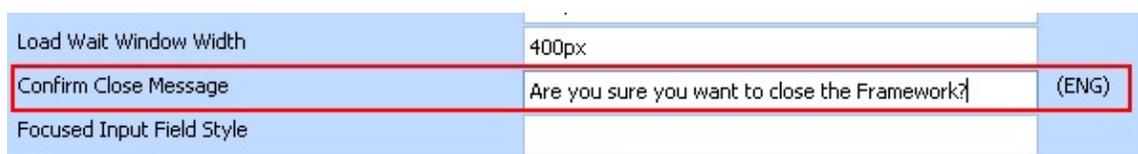
Encrypt Framework Design XML

The Framework design XML can be saved in an encrypted form. The new option [Encrypt XML Files](#) can be found on the Framework Details tab of Framework Properties:



Confirm Close Message

You can now set a confirmation message when end-users attempt to close the Framework. Use the [Close Confirmation Message](#) option on the Web/RAMP Details tab to enter the message which will be displayed before the Framework is closed:



New features in EPC 839 version of the Framework

This section outlines features that were new in EPC839 version of the Framework:

VLF.NET Snap-in Components

You can now create Framework filters and command handlers as .NET components using Visual LANSA Framework .NET SDK.

For more information refer to the Visual LANSA Framework Snap-in Components Guide.

RAMP-TS

A new RAMP product RAMP-TS is now available for modernizing 5250 applications.

It is released for VLF.WIN, VLF.WEB and VLF.NET.

For more information, refer to the [RAMP Guide](#).

Generic Notes and Spool Files Command Handlers

The [Generic Notes Command Handler](#) allows notes to be associated with a business object instance. The notes can be classified, and may have multiple attachments associated with them (eg: MS-Word documents, e-mails, images/photos, etc, etc). All the notes and attachments are permanently stored in database tables on the database server.

The [Generic Spooled Files Command Handler](#) allows the end-user to browse, delete, or change the queue of a spooled file. They can also convert the spooled file to a text file or PDF, which can be copied, emailed, viewed etc.

Export Instance List Contents to Document

End-user can now [Export Instance List Contents](#) to a document. There is no additional effort required from the part of Framework developers. The supported document formats in VLF.NET are:

- Excel
- PDF
- HTML
- CSV

In the Windows environment, templates are not available and the only format currently supported is CSV.

Program Coding Assistant for Command Handler That Sends Data to a Spreadsheet

Force Command Handlers and Filters to Terminate

There is a new Program Coding Assistant for Windows command handlers which generates a command handler that allows the end-user to select from a list of fields and then send the data for the selected fields to a MS-Excel spreadsheet. The end-user can choose to send the data for all entries in the instance list, or only the selected entries.

The Code Assistant is called Send data to MS-Excel as a CSV file.

See the RAMP tutorial [Sending Instance List Contents to Excel](#) for step-by-step instructions.

Alternatively, you can use the option [Allow Instance List to be sent to MS-Excel](#) to automatically send instance list contents to a spreadsheet.

Programmatically Set Focus to a Pane

You can now programmatically cause a pane to be expanded and then become focused using the [avPaneFocus Method](#).

You can add hidden commands to the Framework to allow a user to switch between the main panes without using a mouse.

Set Execution Priority on Server Definitions

The execution priority for super-server jobs can now be set in the server definition. The default priority is 20 however if it is set to a lower figure the super-server job will have a higher priority and receive better service from the CPU. See [Execution Priority](#).

The [Stay Active](#) option for filters and command handlers has been expanded to include new settings, NEVER and DEFAULT.

NEVER indicates that when you move between business objects, any inactive (not visible) command handlers are terminated.

DEFAULT sets the Stay Active option for filters and command handlers to the value set on the new Framework level option, [Stay Active Default for Command Handlers and Filters](#).

Application level tracing has been extended so that you can easily determine the status of filters and command handlers.

Advanced Tutorials

Advanced [Tutorials](#) are now available for Windows, Web and WAM Frameworks.

Reduce the Amount of Memory Your Framework Uses

The new Framework level option [Trim Working Set](#) can be used to reduce the amount of memory your framework uses.

If this option is enabled in a VLF-WIN application the working set of the Windows process

executing the Framework is trimmed:

- When the start up of the Framework is complete
- When the main Framework window has been minimized for approximately 20 seconds.

New Way of Saving Unsaved Changes

There is a new way of [Saving unsaved changes using uQueryCanDeactivate / avNotifyDeactivation](#).

Framework Caption is Now Used as the Web Page Title

A Framework's caption is now used as the Web page title when running as a VLF-WEB application and appears on the IE tab in the web page's language.

Turn Application Tracing on in User Mode

The shipped Demonstration Application has a new Framework level command that activates Application level tracing and can be used in User Mode.

See the source code of reusable part DF_DET46 for an example of how this is done.

Get Icon References

You can use the avFindIcon method to [Get Visual LANSA Framework Icon Reference](#).

Manual Deployment of VLF.NET

In some circumstances a strict security policy can make ClickOnce deployment unviable. A new feature addresses this problem by enabling the network administrator to generate a version of VLF.NET that can be launched locally from, for example, a network share.

[VLF.NET Manual Deployment – An Alternative Way of Deploying VLF.NET](#)

Add Text to the Window or Title Bar Caption

A suffix of any text can now be added to the current window or title bar caption. See the FAQ [Can I change the business object instance caption that appears in the area above my command handlers?](#)

New View Menu Options

New options in the View menu allow the user to move focus to a particular part of the Framework window.

To see how they work, execute for example the Essential Business Object in the Programming Techniques application.

To review new features in previous Framework versions, see:

[New features in EPC 831 version of the Framework](#)

[New features in EPC 826 version of the Framework](#)

[New features in EPC 804 version of the Framework](#)

[New features in EPC 793 version of the Framework](#)

[New features in EPC 785 version of the Framework](#)

New features in EPC 831 version of the Framework

This section outlines new features in EPC831 version of the Framework:

VLF.NET

Any Framework Web Browser application can be compiled as a .NET executable using the [VLF.NET](#) feature. It offers:

- Enhanced security capabilities
- Faster client side application execution
- Visually enhanced and functionally better GUI capabilities
- Near zero deployment by using .NET's integrated web browser deployment technology
-

Developer Interface Enhancements

[Developer's Workbench](#)

[Improved filter and command handler snap-in facility](#)

[Improved Images palette](#)

[Execute the Framework in Prototype Mode Only](#)

[Easy access to latest Demonstration System](#)

[Keep old XML Framework versions organized](#)

User Experience Enhancements

[Visual Themes](#)

[Server connection recovery](#)

[Quick Find Box on the tool bar](#)

Instance List

Improvements

[Show or hide Instance list tool bar buttons](#)

[Programmatic instance list sorting](#)

[Filters can override instance list column headings](#)

[Enable/disable peer objects in instance lists](#)

New Functionality

[Programmatically shrinking and expanding panes in Windows](#)

[General Purpose Document Manager](#)

[Control when an object can be opened in a new](#)

Web Enhancements

[Trace can be disabled for Framework Web applications](#)

[Visual Ids on Web can be hidden](#)

window

Optionally show current business object in window title

Set session values before connection

Test the Activated state of filters and command handlers

Specify your Framework version number

The tabbing order of buttons can be controlled

Improved Focused Input Style

VLF.WEB logon improvements

Improved start up times for DBCS Web applications

Control Web Framework start-up dimensions

Improved XP and Vista fields and buttons in web browser applications

New Limits

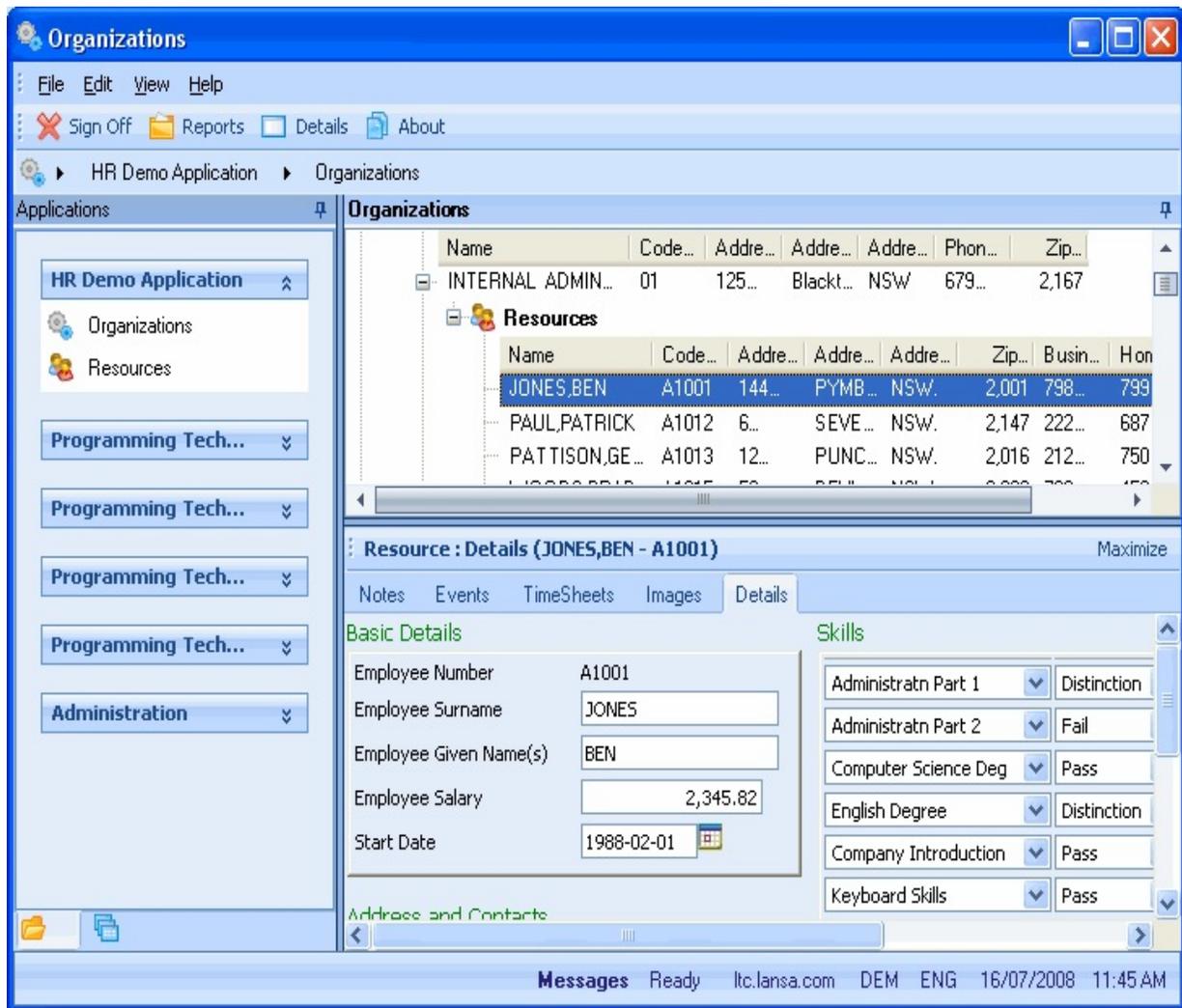
Maximum Number of Application Views increased to 100

Maximum Web Password Length is now 32

Selection Block Size set to 500 by default

VLF.NET

Any Framework Web Browser application can be compiled as a .NET executable.



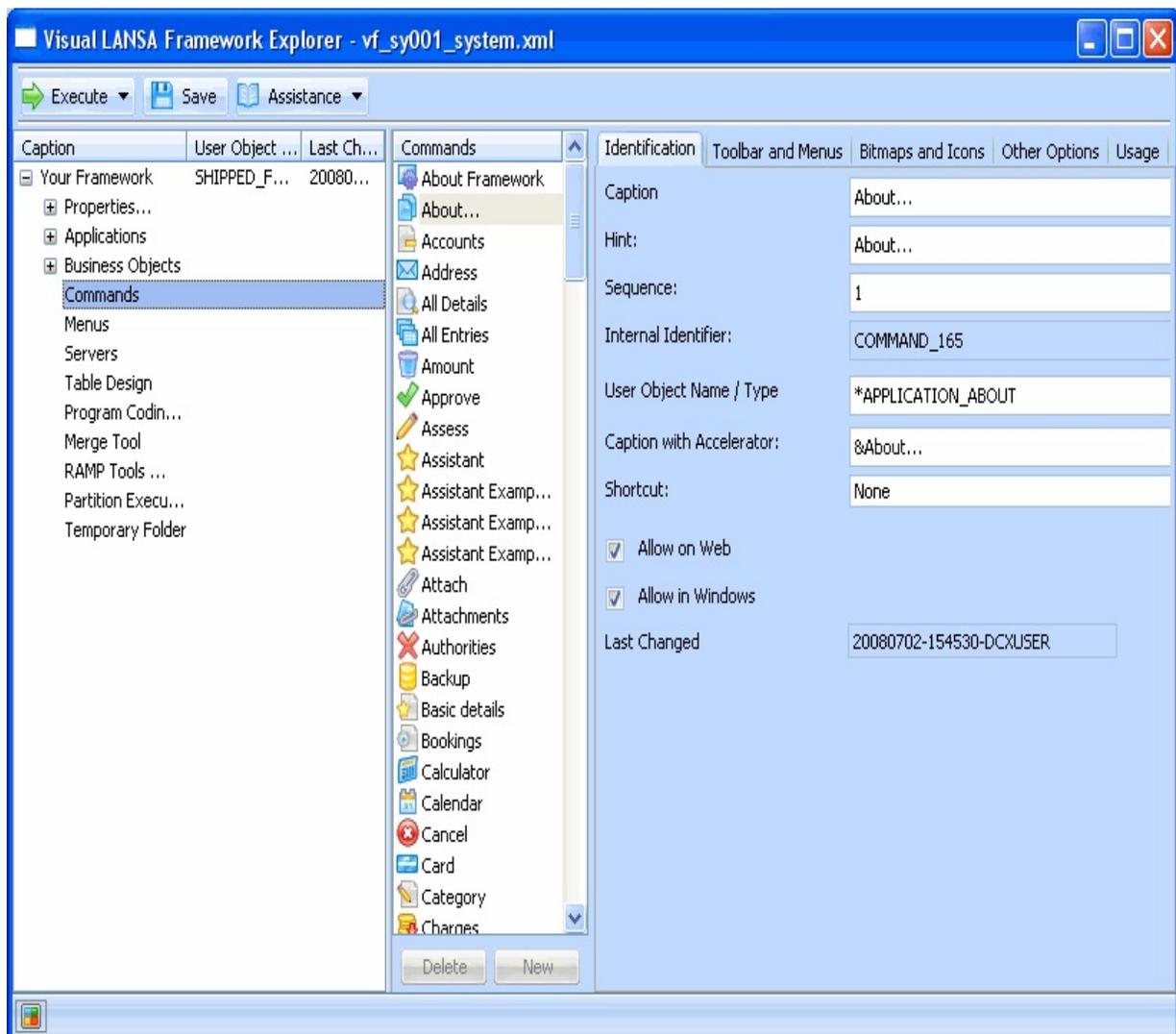
VLF.NET supports all existing and new WAM filters and command handlers and all RAMP screens and scripting.

See [VLF.NET Applications](#).

Developer's Workbench

Developers can now directly access Framework objects and their properties using the Developer's Workbench.

The Workbench has been designed to make it possible to quickly perform actions such as snapping in filters and command handlers and changing properties, without the performance overheads caused by having to display the Framework at the same time.



To start the Developer's Workbench execute form UF_DEVEL.

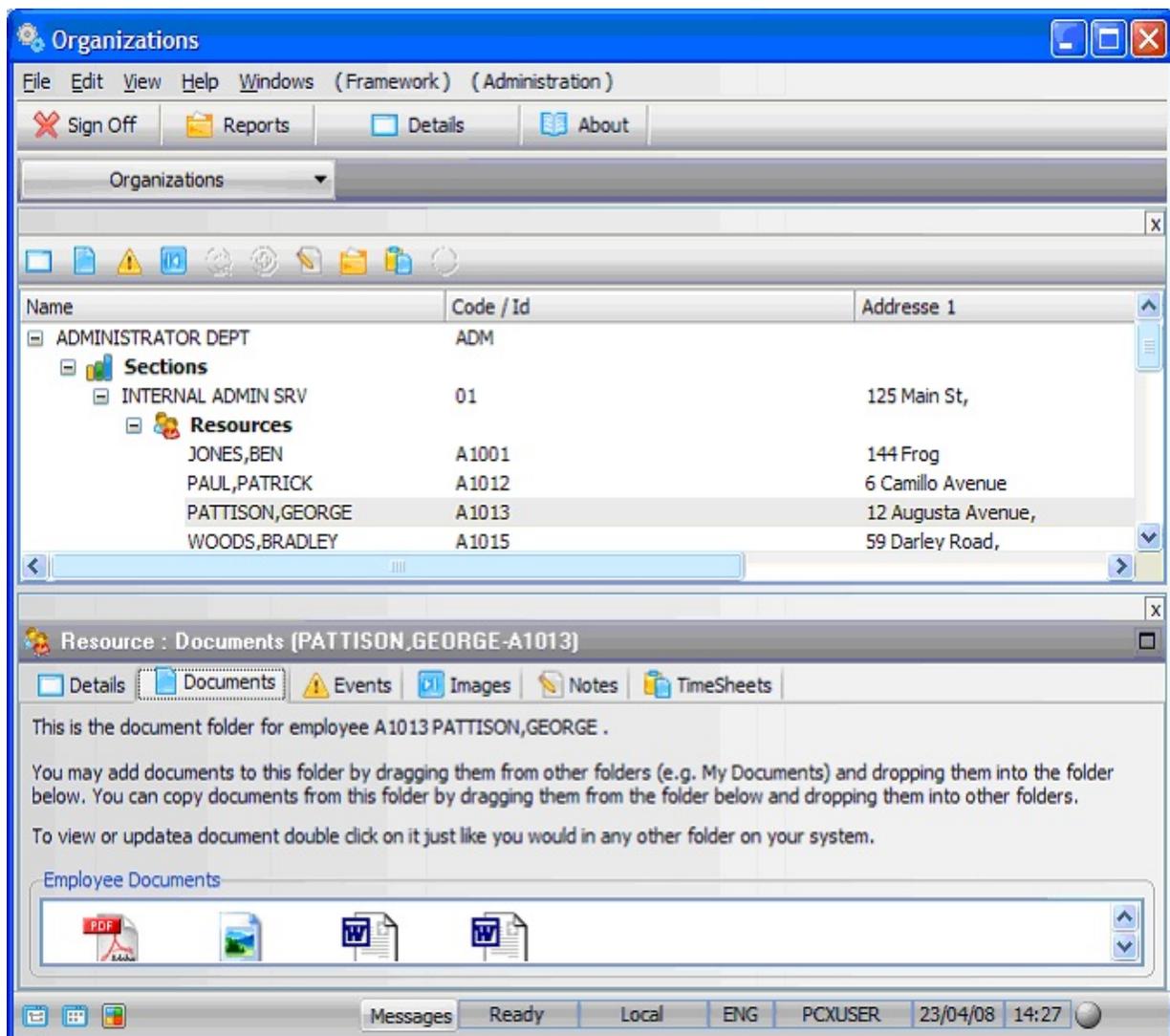
Visual Themes

Visual Themes are an easy-to-activate alternative to visual styles which produce a dramatic improvement in appearance.

Visual themes are available in the Visual LANSA Frameworks executing in Visual LANSA 11.5 or later environments, for Frameworks at level epc826 + hotfixes 050 and 053 (or later).

Just select a theme in --> Framework --> Properties --> Visual Styles tab --> Visual Theme. The appearance of the entire Framework will change to the new theme immediately, including your own command handlers and filters.

You can optionally allow end-users to change their own theme.



Note that it may be necessary on some PCs for the user to shut down and restart the Framework before the theme is fully implemented.

Also see [Overall Theme](#) and [End User can change theme](#).

Trace can be disabled for Framework Web applications

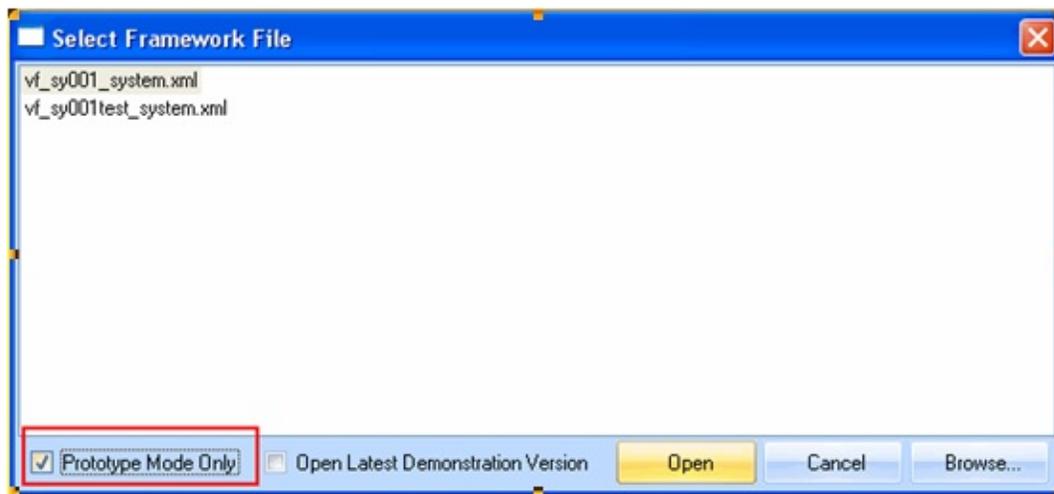
You can now ensure that end-users cannot run tracing when they execute their Framework web applications, even if they add TRACE=Y or TRACE=SYSTEM to their URL.

By setting the field #CHK_TRACE to FALSE in the web sign-on IIP, all tracing after sign-on will be stopped.

The shipped version of this is function UF_SYSBR/UFU0001. See the source code of UFU0001 for more details.

Execute the Framework in Prototype Mode Only

You can now execute the Framework in its original prototype mode by selecting the Prototype Mode Only check box when you select the Framework file.



This option is only available when you execute the Framework as Designer. See [Prototype Only](#).

Improved filter and command handler snap-in facility

The interface for snapping in filters and command handlers has been redesigned for better usability and performance.

The screenshot displays a software interface titled "Command Handler". It is organized into three main sections:

- Windows:** Contains a search bar with the text "DF_DET30" and a magnifying glass icon. Below it is a radio button labeled "Component" and another radio button labeled "Mock Up - RAD-PAD" with a long alphanumeric string.
- Web Browser:** Contains a search bar with "VFLU0352" and a magnifying glass icon, followed by the text "Process: VF_PR003". Below this are radio buttons for "WEBEVENT/Hidden Function", "WAM" (which is selected), "AJAX", and "Mock Up - RAD-PAD". The "AJAX" option has sub-inputs for "HTML Page" and "Module", each with a magnifying glass icon.
- RAMP Destination(s):** A section with the text "Use RAMP Tools to change" and a large empty text input field below it.

The simplified interface allows filters and command handlers to be selected by object type. The search facility can be used to locate the objects by name or description.

RAD-PAD mock-ups can be identified by name and RAMP destination screens are shown.

Improved Images palette

To make it easier to prototype applications, enhancements have been made to the Images Palette including the selection of the images folder and the zooming of image sizes.



Specify your Framework version number

On the (Framework) -> (Properties) -> Identification tab you can now specify your own Framework version number. It consists of four numbers, each of which must be in the range 1 to 9999999.



Your Framework Version Number . . .

Automatically Increment when Saving

Show in Help About Text

The version number is significant when using VLF.NET as it indicates to Internet Explorer that the version of your Framework has been updated on the web server.

See [Your Framework Version Number](#).

Keep old XML Framework versions organized

You can now store your old XML Framework versions in a subfolder to reduce the number of files in the partition execute directory. See the new property in Framework Details --> [Keep Versions in Subfolders](#).

Set session values before connection

For Framework Windows applications that connect to a server, there is a new IIP that allows session values to be set just before the connection to the server occurs. The new IIP is called `avSetBCSessionValues`. See `UF_SYSTM` for details.

Most session values are set by IIP `avSetSessionValues`, but since this occurs after connection to the server, it is not appropriate for some session values (for example session value `PSRR`). The new IIP `avSetBCSessionValues` allows the designer to set the session values that need to be set prior to connection.

Selection Block Size set to 500 by default

The Selection Block Size parameter defines the number of records that are transferred, in one hit, from the server by SELECT commands executed on the client system. Performance is improved by using large block sizes and therefore the default when you define a new server is now 500.

See [Selection Block Size](#).

Server connection recovery

On Windows the Framework can be configured to handle a temporary loss of connection to a server. For example, this might happen when a user's laptop moves out of range of the wireless base station.

The Framework can be configured to check the server connection:

- Before the user moves to the next business object or application
- Before executing a command
- At intervals specified by the designer

If the connection check fails, by default the Framework stops, advises the user and after user confirmation attempts to reconnect.

Alternatively the application designer can write their own server connection test function or perform programmatic connection checking in Framework filters and command handlers.

See [Server Connection Recovery](#)

Programmatic instance list sorting

In the Windows Framework a filter or command handler can now programmatically sort the instance list by a column at run time.

See [Programmatically Sorting the Instance List](#).

Filters can override instance list column headings

In the Windows Framework, filters can be coded to override the instance list column headings at run time. This can be used to make the instance list suit the result of different filter searches.

See [Overriding Instance List Column Headings](#).

Maximum Web Password Length is now 32

The maximum allowable length for passwords can be extended to 32 characters in Web applications. Minimum value allowed is 10, maximum value allowed is 32.

See [Maximum Web Password Length](#).

Optionally show current business object in window title

Use the option [Show Current Business Object in Window Title](#) to force the Framework to always show the current business in the window title.

If this option is unchecked, the Framework caption will be used for all window titles unless the Show the 'Windows' Menu in this Framework option is checked in which case all windows show the current business object as the window title, regardless of this option.

Improved Focused Input Style

A directive, APPEND= can be used with the Focused Input Field Style property of a Framework to cause the style entered in the [Focused Input Field Style](#) property to be appended to the existing style of the input field instead of replacing it.

Improved XP and Vista fields and buttons in web browser applications

Framework fields and buttons in XP and Vista now look like the native ones.

Show or hide Instance list tool bar buttons

You can select or unselect the [Show on Instance List Tool Bar](#) option to control whether a button for a command should appear on the instance list tool bar associated with the instance list.

Note that if child or parent business objects have a reference to the same command definition, a button for these references may still appear.

General Purpose Document Manager

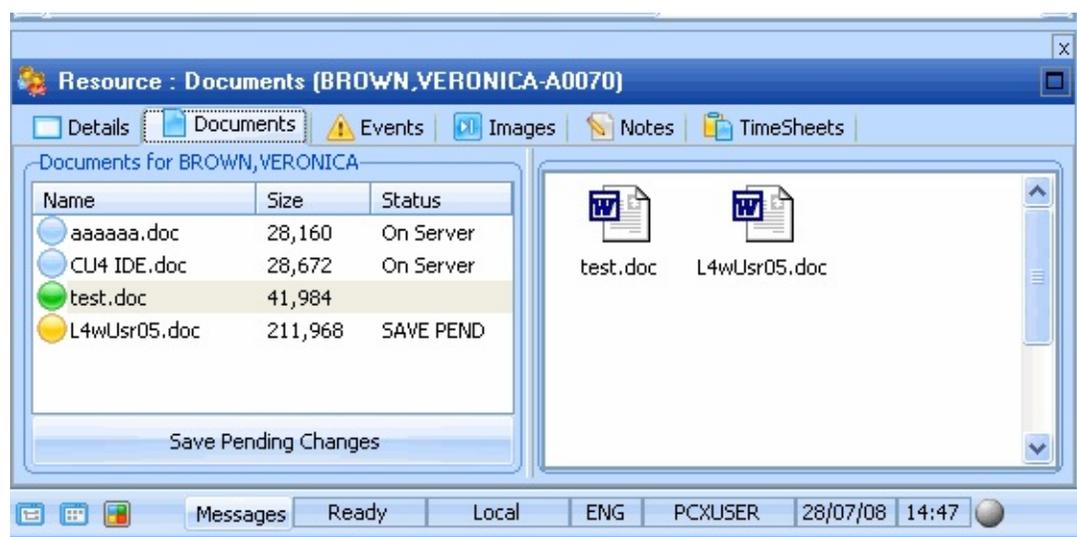
Use the shipped command DX_DOCS to manage a set of user defined documents. The documents are stored in a shipped database file called DXDOCS. You can snap this command to any Framework object and it will be instantly functional. Users are presented with a standard Windows folder browser and a list view initially showing existing documents for the selected object. An icon next to the list view entry indicates the status of the document. The end-user can add documents by dragging and dropping or by cutting and pasting them to the right-hand side of the Document Manager. Documents can be managed using the context menu.

Entries with light blue icons indicate a stored document pending download. Download the document by clicking on the list view entry and then edit or delete it.

Entries with green icons indicate documents that have been downloaded into a local temporary folder.

Entries with yellow icons indicate documents that do not exist in the database. For example documents that have been dragged into the folder browser. Or they can be documents the contents of which have been edited since the time they were downloaded.

Entries with a red cross are documents marked for deletion. For example, downloaded documents that have been deleted or dragged out of the folder browser.



See the shipped Documents command handler for the Resources business object.

Maximum Number of Application Views increased to 100

You can define up to 100 Views for an application. However, more than 10 is considered excessive and will affect the amount of space available to show the views and business objects belonging to them. If more than 10 are required you should restrict the Navigation View Pane to Tree View.

See [Optionally Group Business Objects into Application Views](#) .

Quick Find Box on the tool bar

When the Quick Find feature is enabled, a search field like this appears on the right-hand side of the Framework tool bar like this:



When a user clicks in this field a list of the business objects they have recently been using is displayed. The list is ordered from most recent to least recent:



The user may select a business object from the list by clicking on it. This will immediately switch them to the business object.

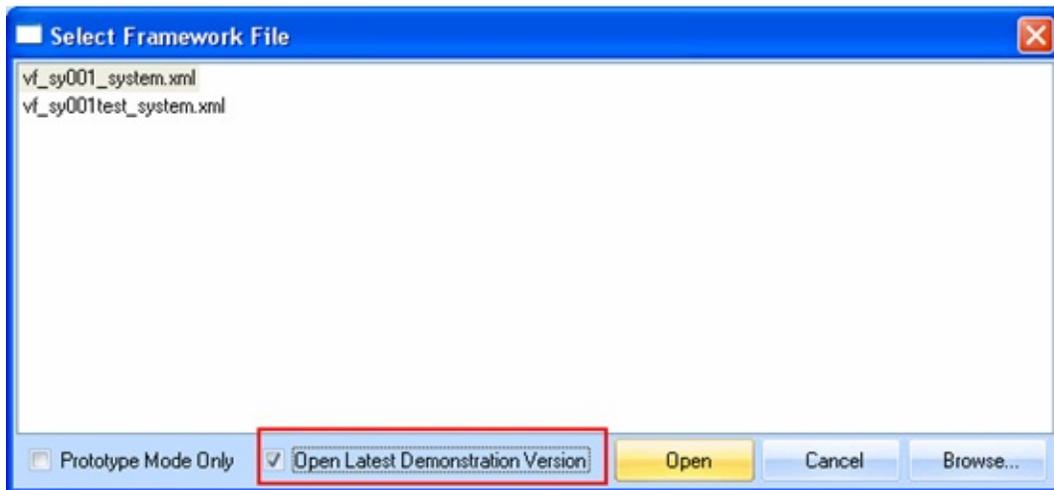
They may also type in a search value. As they type the set of authorized business objects captions is scanned and presented to them. By clicking on one of the business objects presented they will be immediately switched to it:



See [Allow Search/Recently Used Limit](#) and [Search Field Width](#).

Easy access to latest Demonstration System

When starting up the Framework as a Designer, you can launch the latest Demonstration system by selecting the option Open Latest Demonstration Version and clicking the Open button.



See [Open Latest Demonstration System](#).

Enable/disable peer objects in instance lists

The [Enable Peers when Selected](#) option indicates whether any appropriate peer objects' commands should be enabled when this business object is selected.

This option is selected by default.

The tabbing order of buttons can be controlled

The tabbing order of buttons can now be controlled in Web Framework applications.

In WAMs the order is set using the `avSetButton` method (see [UB_xxxxx User Buttons](#)).

Control when an object can be opened in a new window

You have now more control over whether an object can be opened in a new window. The options for [Allow this Object to be Opened in a New Window](#) are:

- | | |
|---------------------------|---|
| Never | This object cannot be opened in another window by an end-user. |
| Manually | This object can be manually opened in another window by an end-user. They do this by using the Window -> Open in a New Window main or popup menu options. |
| Automatically | This object should be opened in a new window automatically when it is selected in the navigation pane. In effect this means that this object will always operate in a separate window to the one it is launched from. |
| Automatically or Manually | This object should be opened in a new window automatically when it is selected in the navigation pane, and, it can be manually opened by an end-user. |

Note: None of these options have any impact on the ability of developers to programmatically open new windows by using the `avShowWindow` method (see [Programmatically Creating and Managing Windows](#)).

Improved start up times for DBCS Web applications

The way the start up DBCS information is exchanged with the server when a VLF.WEB application executes the first business object has been improved. The initial information exchange should now complete in at most 2 exchanges.

Control Web Framework start-up dimensions

For the web Framework, designers now have some control over the relative dimensions of the main Framework panels at start up. (Navigation Pane, Filter Panel, Instance List Panel, and Command Handler panel)

See [Web Initial Filter Pane height \(%\)](#), [Web Initial Filter Pane width \(% of right panel\)](#) and [Web Initial Navigation Pane width \(%\)](#).

Visual Ids on Web can be hidden

In Framework web applications, if a business object has Visual ID 1 or 2 with a sequence of zero or blank, the visual id columns are now hidden.

Programmatically shrinking and expanding panes in Windows

In the Windows Framework a filter or command handler can now programmatically shrink the filter pane, the command handler pane, the instance list, or the navigate pane, and can also re-expand them.

Command handlers can also programmatically expand themselves to occupy the space used by the filters and instance list (Maximize), and restore themselves back to their original size.

See [Expanding, Shrinking and Focusing Panes](#).

Test the Activated state of filters and command handlers

Typically you use the Activated state of filters to test whether signalled events should be ignored.

avFilterActivated Property

This property contains strings TRUE or FALSE and it allows logic in a filter to test whether it is currently in an activated state.

A filter is activated if it is in a state where a user may be able to interact with it.

A filter in a minimized, hidden or deactivated window may still be considered to be activated.

avHandlerActivated Property

This property contains strings TRUE or FALSE and it allows logic in a command handler to test whether it is currently in an activated state.

A command handler is activated if it is in a state where a user may be able to interact with it.

A command handler in a minimized, hidden or deactivated window may still be considered to be activated.

VLF.WEB logon improvements

VLF.WEB application logon credentials are verified by a customer-specified LANSAs function. The shipped example logon verification function is named UFU0001.

The shipped UFU0001 now contains coding and comments that demonstrate how to:

- Validate logon credentials directly against an System i user profile's details.
- Provide specific error messages indicating exactly why logon credentials were rejected.

To accompany these changes sample 3GL programs and their source code are installed by EPC831.

The source code file is named UF_3GSRC. It contains the source code for any 3GL programs that UFU0001 may need to call if you decide to activate this new feature.

Remember that the shipped UFU0001, and any 3GL programs or source code provided, are samples only. You should create your own versions to prevent loss or behavioural change during future VLF upgrades. See the source code of function UFU0001 in process UF_SYSBR for more details.

New features in EPC 826 version of the Framework

This section outlines new features in EPC 826 version of the Framework:

Web

Open and save your Framework web applications before executing them to create new .HTM and .JS files!

You can now create [Framework-AJAX Applications](#) to achieve optimal web performance with functionality close to that of a Windows rich client application.

See also [Enable Framework for AJAX style applications](#).

Usability

In Windows applications the main Framework panes can be shrunk and expanded to allow more efficient use of screen real estate. Use the [Allow Panes to be Shrunk and Expanded](#) option to enable this functionality.

You can use the new [Focused Input Field Style](#) property to highlight the input field that has focus with a different style in Framework web applications.

Demonstration Application

[The Demonstration Application](#) has been significantly revised and updated.

To use it you need to open the new version of the shipped Framework definition in file VF_Sy001_System_LastShipped.XML.

You need to remove the old demonstration application from any existing Framework versions as it may no longer function correctly.

Security

In web applications it is now possible to replace the Framework's security model by specifying your own avCheckAuth method (this has always been an option in Windows applications).

To do this you need to supply a LANSa function and enroll it into the Framework. Refer to shipped example function UFU0016 in process UF_SYsBR for more details. Also see [IIP – Function to return web user authorities](#).

Mini-filters

You can now specify the mini-filter as a panel to control all its display attributes. See [AvMiniFilterPanel](#).

Web Deployment

It is now possible to use the Administrator interface in a deployed Framework web application to create

For example you can add combo boxes, drop downs, check boxes, and do instant editing on the panel.

Commands

Commands (and command tabs) can now be conditionally disabled or hidden depending on business object [SubTypes](#).

Also, you can use the [Enable Child when Parent Selected](#) and [Enable Parent when Child Selected](#) properties to control command availability.

An option has been added to instance commands to allow a command never to be the default, even when the user used it on a previous instance (see [Default Command](#)).

Commands can now be executed without any user interface at all, that is, hidden from the user. See [Execute as Hidden Command](#) and [Hidden Command Handler Anatomy](#).

users and groups, set authorities and custom property values.

This means that the Windows administrator components that were previously required in web deployment packages can now be omitted. See [Creating Web Interface for Maintaining Users and Authorities](#).

Instance Lists

[Instance Lists with Different Types of Objects](#) are now also available in Framework web applications.

Peer business objects are now shown with their icon to make it easier to distinguish different types of objects.

You can [Work with Hidden Child Objects](#) by double-clicking an instance list entry.

Instance lists now have a toolbar. You can set the [Instance List Tool Bar Location](#) and the [Instance List Tool Bar Height or Width](#) in the Instance List/Relations tab of the business object.

ActiveX instance lists are no longer supported.

In Windows applications, Instance lists that are saved and restored between VLF sessions now include the user object name/type of any current super-server connection in the saved file names. This means that different versions of a saved instance list can co-exist for example for Server1, Server2 and a Local DBMS.

Command Usage Tab

More details about the usage of command definitions are available on the Command Usage Tab:

- Default Command
- Hidden
- RAMP Destination

Performance

This version of the guide includes detailed information about [Assessing Performance in Framework Web Applications](#).

HTML Startup Page

You can specify an HTML page as the startup page for the Framework or an individual application. You can provide links to other resources from this page. See [The Demonstration Application](#) for an example.

Improved problem analysis

The Framework web application trace facility (+Trace=Y on URL) now displays the details of HTML pages arriving at the browser.

If an application detects a bad response from the web server, the user is given an option to analyze the nature of the error and display the problem HTML page. All details can be printed or copied to clipboard making it easier for end-users to provide helpful error information.

Position Menu Option can be Disabled

The Position menu option can now be turned off by designers so that users will not be able to change the relative position of any of the Framework display panes.

New features in EPC 804 version of the Framework

This section outlines new features in EPC 804 version of the Framework:

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 LANSAs that access 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 LANSAs within the same LANSAs 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 LANSAs system.

Usability

Multiple [Framework Windows](#) can be opened at the same time to enhance end-user productivity.

You can create a [Favorites Folder](#) using the Contains Favorites property of an application.

[Navigation pane view buttons have changed](#) so that users can navigate to the required view in a single click.

The [Command Usage tab](#) shows where a selected command is used.

Instance Lists

[Improved Application Tracing for Relationship Handlers](#) are provided to make finding problems in your relationship handlers easier.

Virtual Clipboard

You can use [New Virtual Clipboard Control Options](#) to set clipboard default content and to clear the clipboard.

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.

For Non-English Systems

[New default translation tables for Framework server definitions](#) using RDMLX partitions and connections have been changed to *JOB/*JOB for better default translation results.

There are new text strings that need to be

Deployment

are available. The detailed check lists guide the user in planning the deployment, packaging the material and installing the package.

Imbedded Interface Points

There are [New UF_SYSTEM IIP \(Imbedded Interface Point\) methods that you can override](#).

These methods are invoked when Framework windows are opened and ready for work, or when they are closing.

Look and Feel

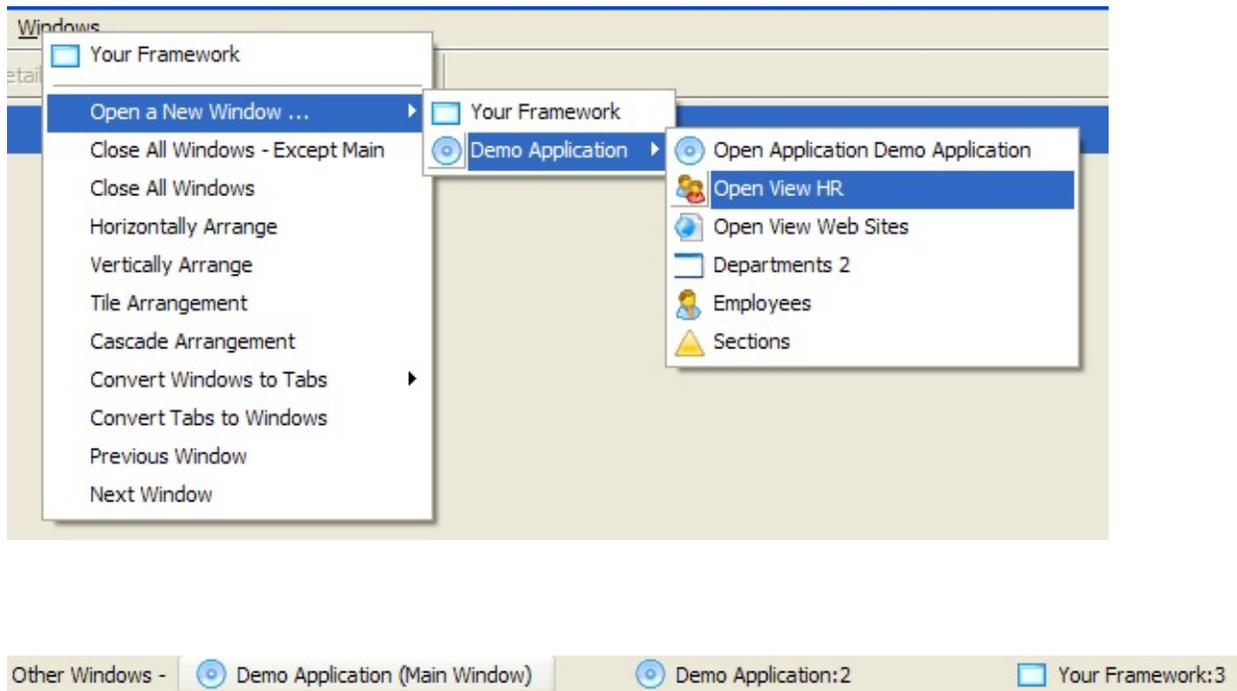
[Applications in tree navigation pane are shown in bold](#)

[Instance List can optionally be displayed with alternate row color](#)

translated. Translation of end-user visible
text is carried out using function
UFU0003.

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

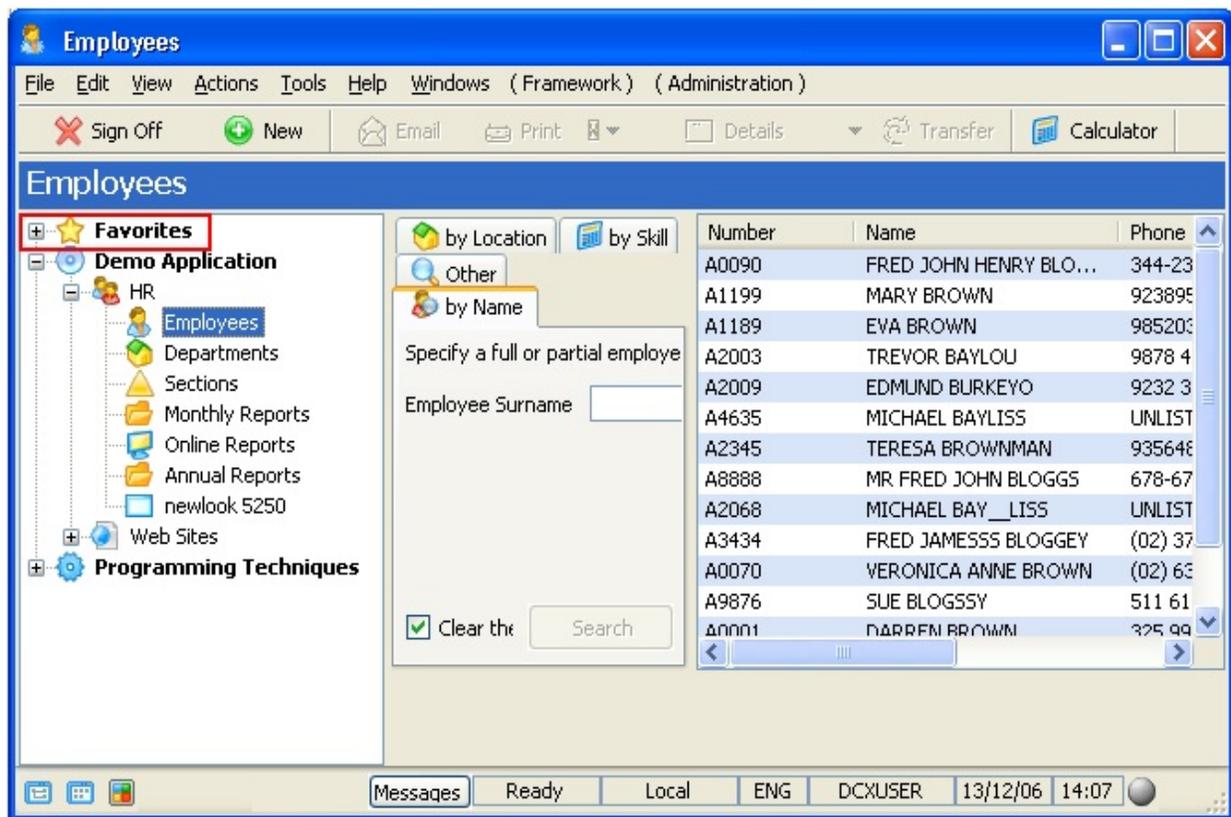
For more information see [Framework Windows Management](#).

Favorites Folder

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:

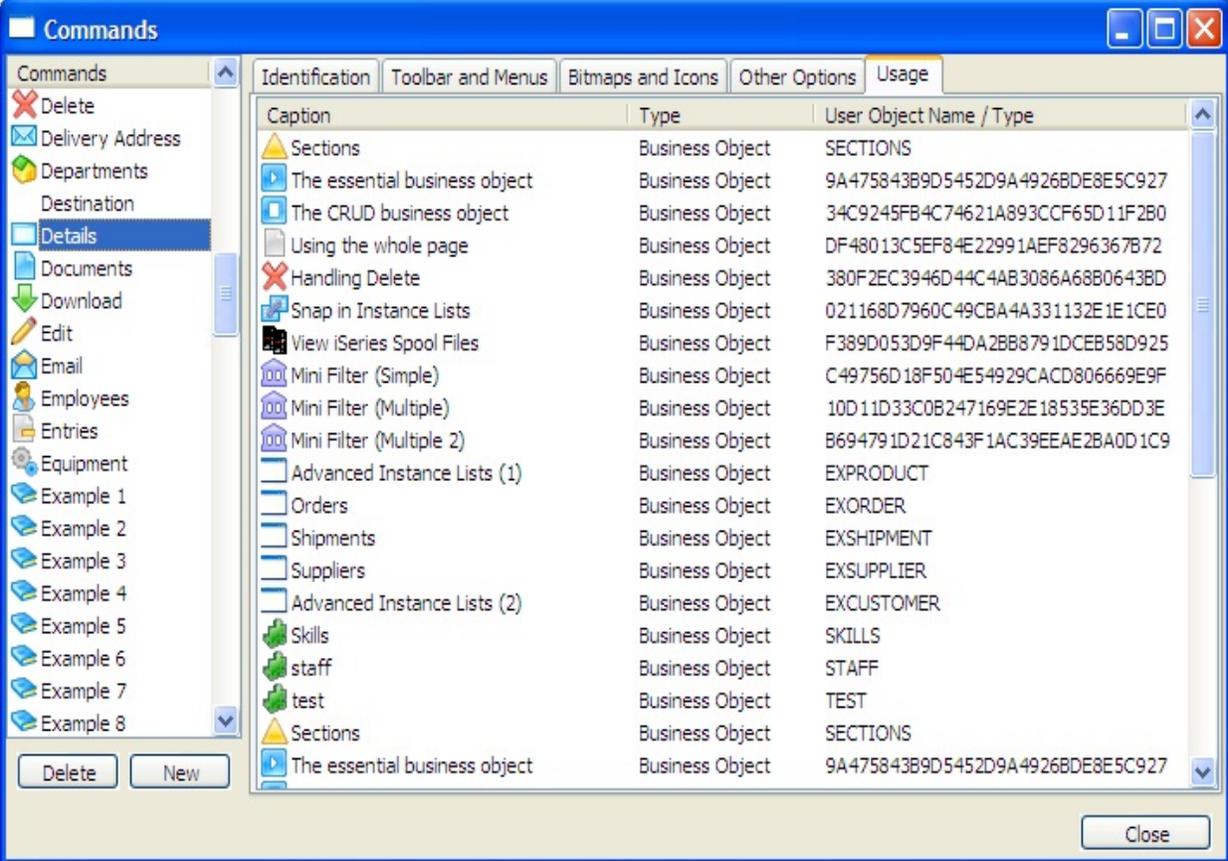
Property	Value
Identification	Bitmaps and Icons Startup Visual Styles Commands Enabled Command Display Help About
Caption	Favorites (ENG)
Hint:	(ENG)
Sequence:	1
Internal Identifier:	408E8663F5934B538D3E702234FA9CE2
Unique Identifier:	476
User Object Name / Type	FAVORITES Verify Name
<input type="checkbox"/> Restricted Access	
<input type="checkbox"/> Allow on Web	
<input checked="" type="checkbox"/> Allow in Windows	
<input checked="" type="checkbox"/> Contains Favorites	
Last Changed	20061108-124350-DCXUSER

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

This feature is available in **Windows** only.

Command Usage tab

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



The screenshot shows a window titled "Commands" with a "Usage" tab selected. The window is divided into a left sidebar and a main content area. The sidebar contains a list of commands, with "Details" selected. The main content area displays a table with three columns: "Caption", "Type", and "User Object Name / Type". The table lists various business objects and their associated user object names.

Caption	Type	User Object Name / Type
Sections	Business Object	SECTIONS
The essential business object	Business Object	9A475843B9D5452D9A4926BDE8E5C927
The CRUD business object	Business Object	34C9245FB4C74621A893CCF65D11F2B0
Using the whole page	Business Object	DF48013C5EF84E22991AEF8296367B72
Handling Delete	Business Object	380F2EC3946D44C4AB3086A68B0643BD
Snap in Instance Lists	Business Object	021168D7960C49CBA4A331132E1E1CE0
View iSeries Spool Files	Business Object	F389D053D9F44DA2BB8791DCEB58D925
Mini Filter (Simple)	Business Object	C49756D18F504E54929CACD806669E9F
Mini Filter (Multiple)	Business Object	10D11D33C0B247169E2E18535E36DD3E
Mini Filter (Multiple 2)	Business Object	B694791D21C843F1AC39EEAE2BA0D1C9
Advanced Instance Lists (1)	Business Object	EXPRODUCT
Orders	Business Object	EXORDER
Shipments	Business Object	EXSHIPMENT
Suppliers	Business Object	EXSUPPLIER
Advanced Instance Lists (2)	Business Object	EXCUSTOMER
Skills	Business Object	SKILLS
staff	Business Object	STAFF
test	Business Object	TEST
Sections	Business Object	SECTIONS
The essential business object	Business Object	9A475843B9D5452D9A4926BDE8E5C927

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 from the VLF.WIN and VLF.NET frameworks.

(The VLF.WEB framework remains the same):



In the VLF.WIN framework the buttons have been replaced by three small icons on the left the status bar like this:



In the .Net framework the buttons have been replaced by two small icons on the bottom left of the navigation pane like this:



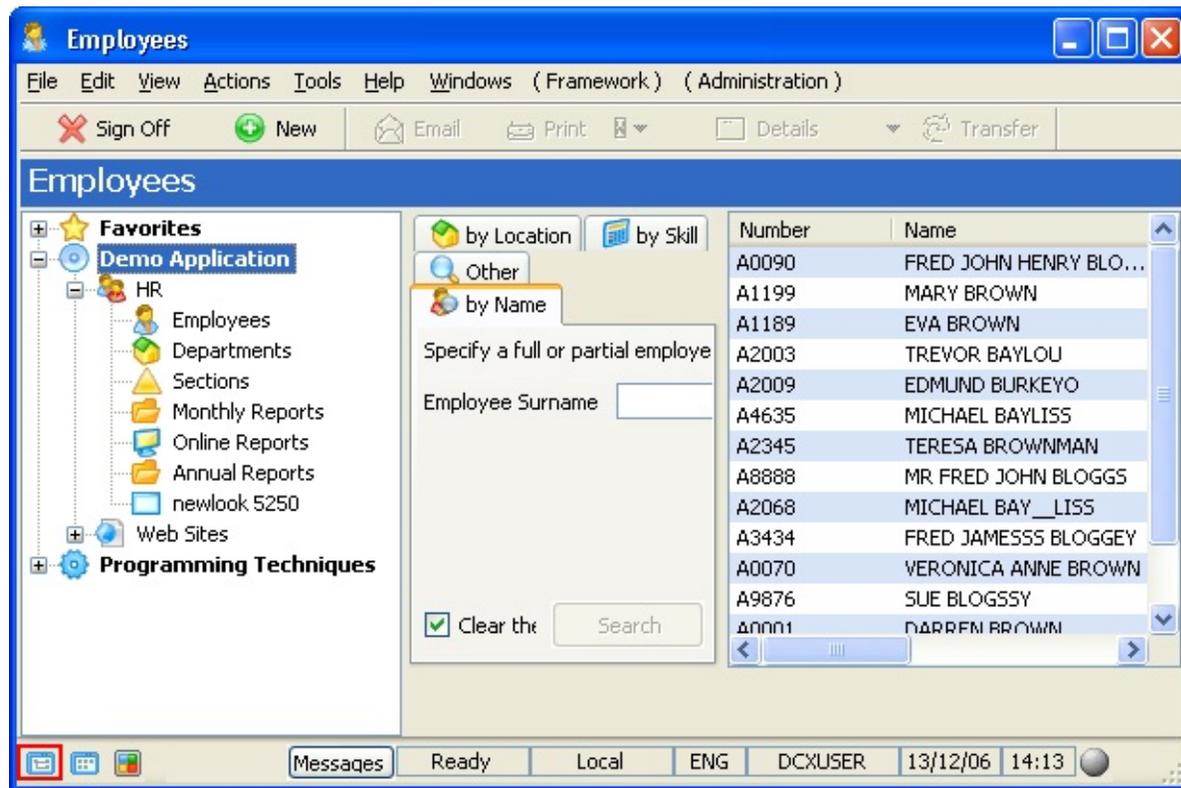
The icons represent the tree, list and drop-down button options. The user can now navigate to the required view in a single click.

[Tree view](#)

[List view](#)

[Drop-down button](#)

Tree view



List view

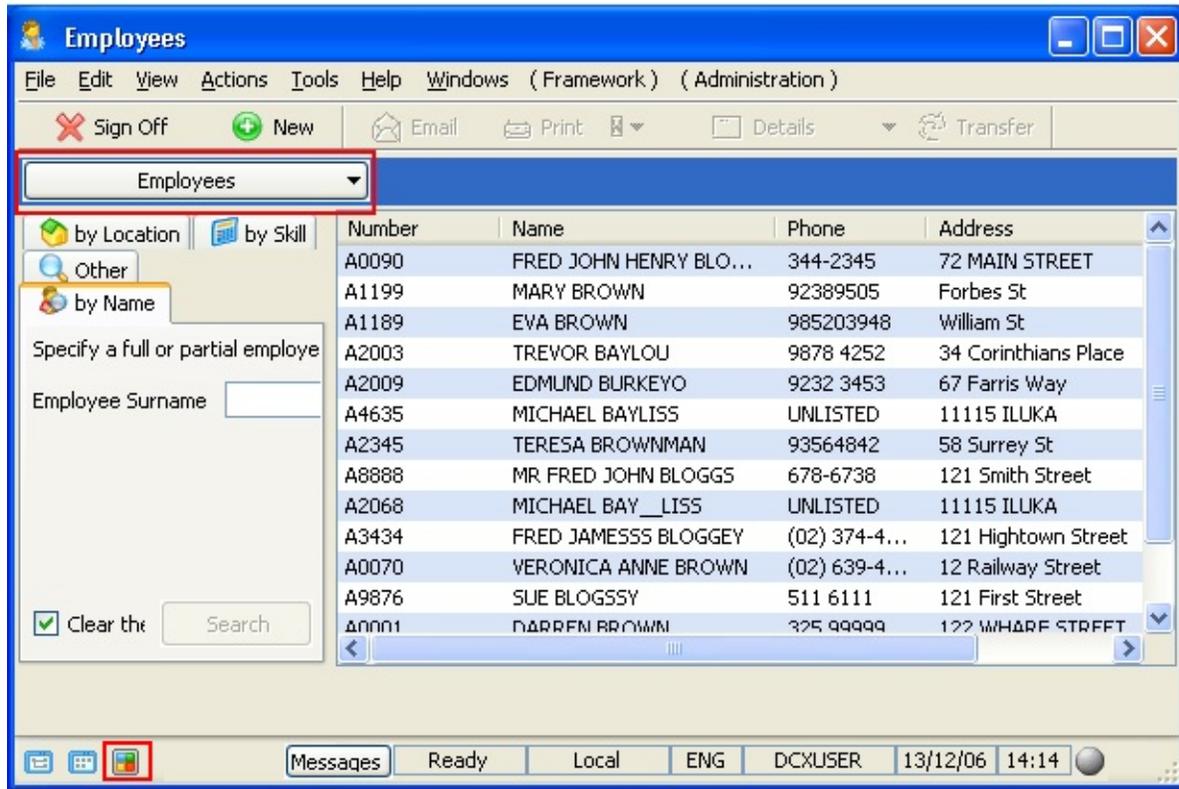
The screenshot shows a web-based application window titled "Employees". The interface includes a menu bar (File, Edit, View, Actions, Tools, Help, Windows (Framework) (Administration)), a toolbar (Sign Off, New, Email, Print, Details, Transfer), and a main content area. On the left, there is a navigation pane with "Favorites" (star icon), "Demo Application" (CD icon), and "Programming Techniques" (gear icon). Below this are "HR" and "Web Sites" buttons. The main area has a search section with tabs for "by Location", "by Skill", "Other", and "by Name". The "by Name" tab is active, showing a search box labeled "Specify a full or partial employee" and "Employee Surname" with a "Search" button and a "Clear this" checkbox. To the right is a table of employee records.

Number	Name
A0090	FRED JOHN HENRY BLO...
A1199	MARY BROWN
A1189	EVA BROWN
A2003	TREVOR BAYLOU
A2009	EDMUND BURKEYO
A4635	MICHAEL BAYLISS
A2345	TERESA BROWNMAN
A8888	MR FRED JOHN BLOGGS
A2068	MICHAEL BAY__LISS
A3434	FRED JAMESSS BLOGGEY
A0070	VERONICA ANNE BROWN
A9876	SUE BLOGSSY
A0001	DARRRN BROWN

The status bar at the bottom shows "Messages", "Ready", "Local", "ENG", "DCXUSER", "13/12/06", and "14:14".

Drop-down button

The drop-down button view hides the navigation pane and gives access to applications and business objects from a drop-down button:



Application designers can still prevent this option from being used from the Framework properties and specify that a specific view (list, tree or toolbar) is to be used for all end-users.

Applications in tree navigation pane are shown in bold

The screenshot shows the 'Employees' application window. The title bar reads 'Employees' and the menu bar includes 'File', 'Edit', 'View', 'Actions', 'Tools', 'Help', 'Windows (Framework)', and '(Administration)'. The toolbar contains 'Sign Off', 'New', 'Email', 'Print', 'Details', and 'Transfer'. The main area is divided into three sections:

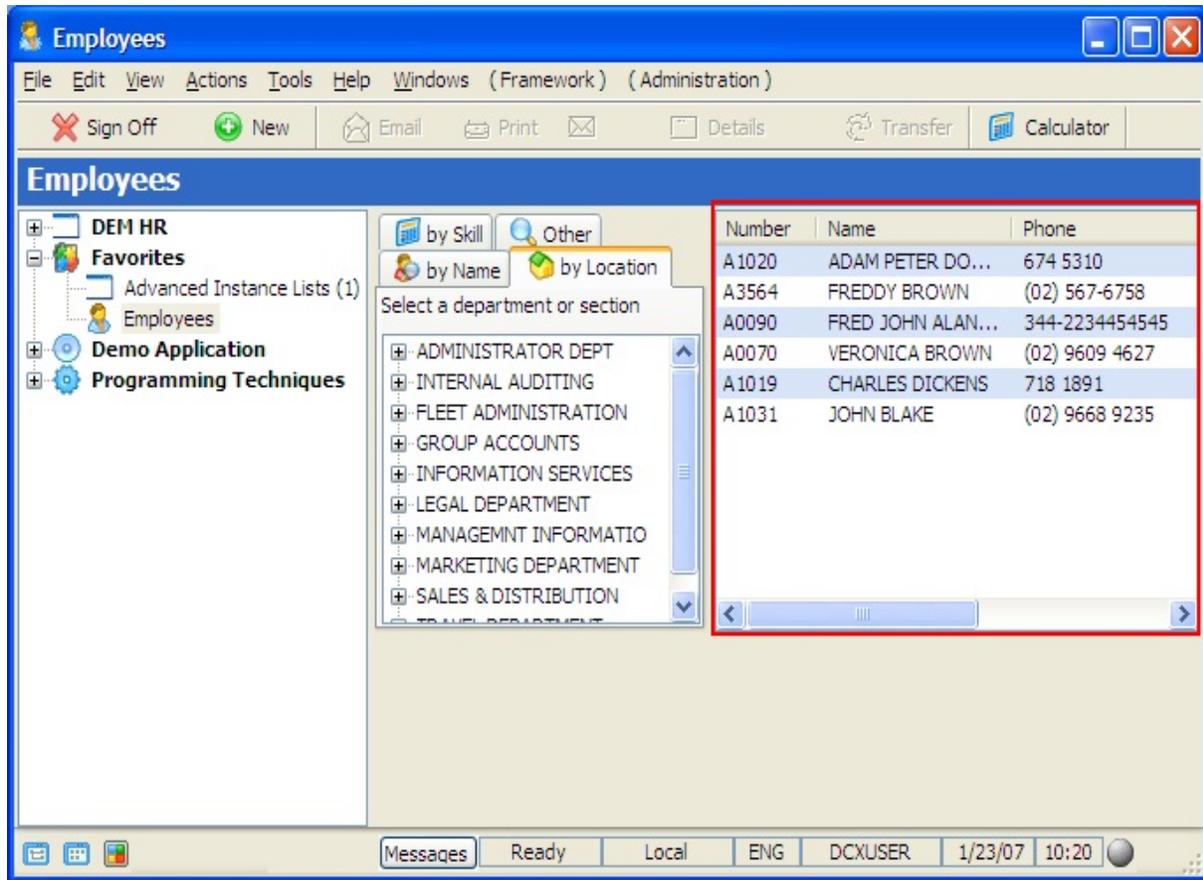
- Tree Navigation Pane (left):** Shows a hierarchy of folders. 'Demo Application' and 'Programming Techniques' are bolded. Under 'Demo Application', 'Employees' is also bolded. Other folders include 'Departments', 'Sections', 'Monthly Reports', 'Online Reports', 'Annual Reports', 'newlook 5250', and 'Web Sites'.
- Search Panel (middle):** Offers search options: 'by Location', 'by Skill', 'Other', and 'by Name'. It includes a text input field for 'Employee Surname' and a 'Search' button.
- Employee List (right):** A table with columns 'Number' and 'Name'. The list contains the following entries:

Number	Name
A0090	FRED JOHN HENRY BLO...
A1199	MARY BROWN
A1189	EVA BROWN
A2003	TREVOR BAYLOU
A2009	EDMUND BURKEYO
A4635	MICHAEL BAYLISS
A2345	TERESA BROWNMAN
A8888	MR. FRED JOHN BLOGGS
A2068	MICHAEL BAY__LISS
A3434	FRED JAMESSS BLOGGEY
A0070	VERONICA ANNE BROWN
A9876	SUE BLOGSSY
A0001	DARRRN BROWN

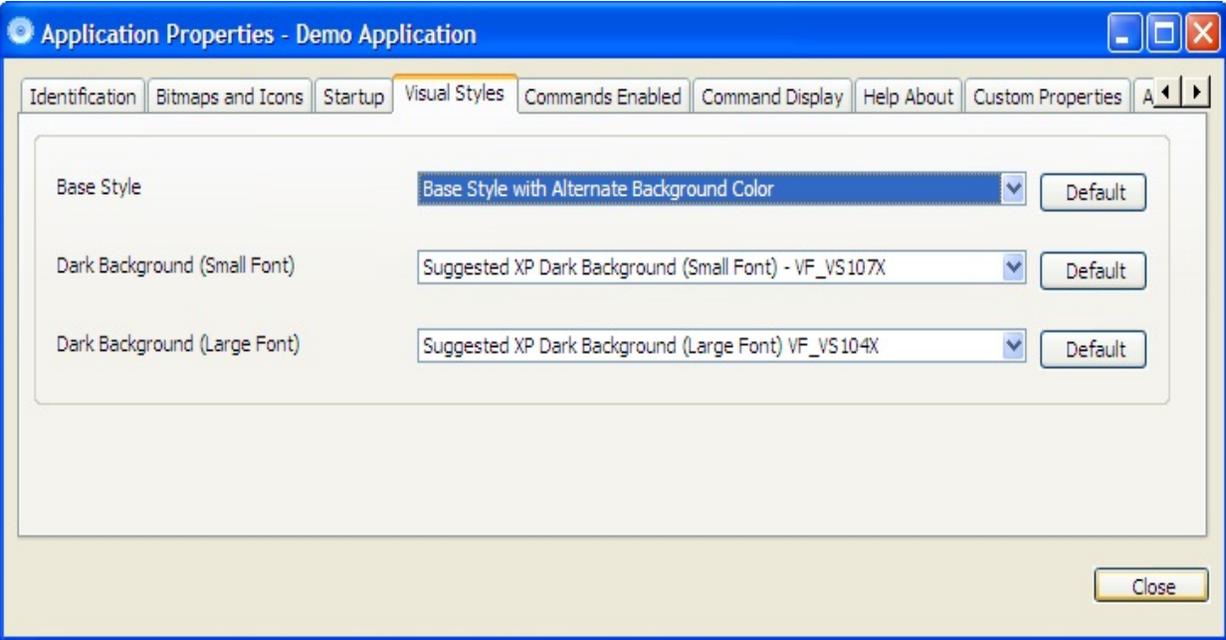
The status bar at the bottom shows 'Messages', 'Ready', 'Local', 'ENG', 'DCXUSER', '13/12/06', and '14:13'.

Instance List can optionally be displayed with alternate row color

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



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



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 LANSAP function UFU0003 (or your equivalent). A new version of UFU0003 has been shipped.

Review UFU0003 (in process UF_SYSTR) 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.

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.

New UF_SYSTM IIP (Imbedded Interface Point) methods that you can override

New IIP methods in UF_SYSTM (or equivalent) are now symmetrically invoked when the main Framework window or a secondary Framework window is opened and ready for work, or when they are closing.

The new methods are called avMAINWindowReady, avSECONDWindowReady, avCloseMAINWindow, avCloseSECONDWindow

The new avXXXXXWindowReady methods are invoked when the window has been opened, the user has been logged on and connected to any server, and 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.

See [Windows and Imbedded Interface Points](#).

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.

For more information see [Persistence, Resetting and Deploying in Windows Applications](#).

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.

New features in EPC 793 version of the Framework

This section outlines new features in EPC 793 version of the Framework:

Performance

The start-up times for Framework applications have been substantially improved in this version:

[Faster Framework web start-up times including RAMP](#)

[Faster Framework Windows start-up times including RAMP](#)

[Improved Framework web application instance list performance](#)

[The Preload Framework Images option has been removed](#)

However, the changes made to accomplish this mean that:

[You Need to Regenerate Your Web Browser and Javascript Files](#)

[The Web Browser Application Load Window has Changed](#)

More Flexible Framework Locking

[New PROGRAM_EXIT option for Framework locking](#)

Instance Lists

[Improved Framework Windows instance list handling](#)

[Improved documentation for instance list processing and options](#)

If you are using the shipped instance list handler DFRELB1 or DFREL01 to handle hierarchical instance lists, you need to recompile them.

Interface Improvements

[New Show in Menu when Disabled option](#)

[New method #Com_Owner.avShowMessages](#)

To review new features in previous Framework versions, see:

[New features in EPC 785 version of the Framework](#)

You Need to Regenerate Your Web Browser and Javascript Files

The main HTML and JavaScript files used for web browser based applications has been changed.

You will need to (re)generate these before attempting to execute any existing web browser based Framework. To do this open your Framework as a designer, make a dummy change to it, then save it again and upload the resulting HTML and JavaScript files to your web server.

Where you have a deployed application you need to regenerate your deployed Framework HTML and JavaScript files and deploy them ready for installation at the same time as you upgrade your deployed VLF environment.

The Web Browser Application Load Window has Changed

If you use the "Load Window" that appears when a web browser application is starting over the internet you may need to specify a height and width for the updated version of the load window.

The existing property "Web Load Style" has been removed and replaced by individual window height and width fields.

Faster Framework web start-up times including RAMP

The HTML and JavaScript files that are generated for Framework Web applications have been significantly changed.

By using an 'unrolling' technique for your Framework definition, the start up times for Framework web applications have been further significantly reduced. As a result of these changes the XML file that defines a Framework no longer needs to be deployed to end-user environments.

Faster Framework Windows start-up times including RAMP

Framework Windows applications now use a new property optimization technique that has reduced the start up times for applications using the end-user entry points UF_EXEC and UF_ADMIN (or equivalent).

Generally, deployed Framework Windows applications should start up faster than before.

Improved Framework web application instance list performance

Instance list handling in Framework web applications has been revised, resulting in significantly better instance list performance.

Improved Framework Windows instance list handling

In Framework Windows applications instance list handling has been improved. It is now possible to dynamically update, insert and delete entries in instance list visualized as tree views without refreshing the entire tree. Refer to the updated documentation for more details of all the new options.

Improved documentation for instance list processing and options

New sections on instance list processing have been added:

[Advanced Instance List Processing](#) shows possible techniques for the centralized handling of instance lists operations using a Scope(*Application) reusable VL part and methods for delegating common tasks to a shared VL reusable component.

[Updating and Deleting Instance List Entries](#) and [Physical Instance Lists](#) sections describe how to dynamically update, insert and delete entries in instance list visualized as tree views without refreshing the entire tree.

[Instance Lists with Different Types of Objects](#) contains more detailed information about creating parent-child instance lists.

New PROGRAM_EXIT option for Framework locking

The Framework locking facility has a new option PROGRAM_EXIT. It is identical to the existing PROGRAM option except that it allows the user to exit/close down the Framework without releasing the lock.

See [Framework Locking Service to Handle Unsaved Changes](#)

New Show in Menu when Disabled option

This new option allows you to indicate whether a disabled command should show (or not) on command menus.

By setting this option off, the cluttering of menus with disabled commands can be reduced.

See [Show When Disabled](#).

New method #Com_Owner.avShowMessages

Filters and command handlers in Framework Windows applications can now programmatically cause the current set of messages to be displayed by invoking the #Com_Owner.avShowMessages method.

It acts exactly as if the user had clicked the Messages button on the status bar. Framework web applications have always been able to do this.

See [Show Messages Service](#).

The Preload Framework Images option has been removed

The Framework web application option Preload Framework Images has been removed. While using this option visually improved a first time Framework users' experience, it was found that subsequent Framework usage may be actually slower when using this option because of the way that Internet Explorer manages image caching.

Now first time users (or users using the Framework after the IE file cache has been cleared) may experience a slight delay and visual disruptions while images are (re)loaded from the server. Subsequent usage, where images are in the IE file cache, will benefit. You can improve this situation even further by implementing configuration settings on your HTTP server so that browser checking for updated images is not performed when an image already exists in your browser cache.

In a System i Apache web server configuration, this is done in the Expires Header of the Http Responses section as shown in this picture. In this example any .gif files would be version checked every 6 months of the file being accessed:

HTTP Responses ?

Error Message Customization **Response Headers**

Expires Header Meta-information Files

Generate Expires HTTP header for responses: Enabled ?

Default expiration time: ?

Expiration time: Seconds ?

After: ?

Set expiration time based on resource MIME type: ?

	MIME file type	Time	After
<i>Example</i>	<i>image/gif</i>	<i>2 Days</i>	<i>Document access by client</i>
<input checked="" type="radio"/>	<input type="text" value="image/gif"/>	<input type="text" value="6"/> Months ?	<input type="text" value="Document access by client"/> ?

Note that there is a difference between the caching of the image and the version

checking. Browsers would usually reuse the cached image. However, they would still compare the date time stamp of the cached image with the server's one. It is this operation that is suppressed for the specified period of time according to this setting.

New features in EPC 785 version of the Framework

Web

Internet Explorer (IE) 7 is now supported as a browser in Framework end-user applications.

Note that at this date IE7 is still a Microsoft beta product and it is not yet supported by the Visual LANSAs Development Environment.

Also see [How can I hide the address and status bars on Framework popup windows when using IE7?](#)

Framework Versioning

[Merge Items from One Framework to Another](#)

[The Way the Demonstration System is Installed has Changed](#)

Programming Tips

[Advanced Enter Key Handling in VL applications](#)

Code Assistant Improvements

Code Assistants can now optionally create Visual LANSAs objects directly into the Visual LANSAs Development Environment, so no more copy/paste between the Code Assistant and the Development Environment is required.

See [Create Component](#)

Performance

Business Object instance list processing is faster in all Visual LANSAs Framework Windows applications.

The Way the Demonstration System is Installed has Changed

Previously a new version of the demonstration system was automatically installed every time you upgraded your Framework. This no longer happens in order to make sure you do not lose any changes you may have made to the demonstration system.

It is, however, recommended you install the latest version of the demonstration system so that you can see how new features work and to ensure there are no incompatibilities between the shipped components and older versions of the demonstration system.

Merge Items from One Framework to Another

You can merge items from one Framework to another using the Merge Tool.

See [How to Merge Items from one Framework to Another](#)

If You Want Your Project to Succeed

You must perform these essential activities if you want your Framework project to succeed:

1. **Build a prototype** and get it reviewed and formally agreed and signed off by both end-user representatives and developers. Users will know what they are getting and developers will know what they need to deliver.

This is a basic form of expectation management and also goes some way towards managing classic IT project problems such as lack of user engagement and scope creep.

2. **Publish a Minimum Supported Configuration Document** – where you formally state define the minimum configuration your solution will viably support, including what servers, client platforms and web browsers your application will support:

- Minimum hardware requirements (see [Computer System Requirements](#))
- Minimum software requirements
- Supported screen resolutions
- Minimum networking capabilities
- Maximum data volumes.

A formal MSC will:

- Inform decisions about the overall solution cost
- Establish the environment required to test the deployment of the solution or any patch/hotfix made to it.
- Raise management's awareness of the risk in implementing a "sub-MSC" solution.

For more information refer to [Application Performance](#).

Regularly performance test to the minimum platform while doing development. Maintain and republish this document during the project.

3. **Publish a Business Value Proposition** – where you formally define the business value of your application, especially where an existing IBM i 5250 application is being modernized or replaced.

Formally state how and why your application will make doing business better and/or faster and/or smarter. If you can't define the application's value proposition in words and pictures it is extremely unlikely you will be able to

implement it in software.

The introduction of visual components by themselves (ie: things like radio buttons, drop downs, menu bars and colour gradients) rarely represent significant business value. Recidivist end-user behaviour is a common sign of low, absent or poorly explained business value in IT projects.

- 4. Make time in your project plan to for deployment and testing.** You would be amazed how often this area is discounted by developers as something that "will only take a few days". This is classic area where IT projects experience cost overruns.

You need to factor in time to design a deployment strategy, implement it and then test all the supported platforms – in addition to your normal application level unit and suite testing. Optimally you should have human and hardware resources solely dedicated to application testing.

Getting Started

Computer System Requirements

Other Requirements

Starting the Framework

Should you use Windows or Web Browser Applications?

Setting Up Your Framework Environment

The Demonstration Application

The Programming Techniques Application

Computer System Requirements

Please refer to the [hardware requirements for Visual LANSA Version 13 SP2](#).

VLF Developers should only use the Windows 100% (smaller) font size for development activities. Where VLF end users may not be using the 100% (smaller) font size the VLF application should be designed, sized and and tested at the required larger font sizes.

For deployed applications the system requirements depend on the size and complexity of your application. This may seem to be a less than useful statement, but realistically only you can sensibly size your own resulting application.

Other Requirements

Software

- LANSA Version 13 SP2 with EPC 132300 or later.
- The shipped LANSA Personnel System data installed (installed by Partition Initialization).
- aXes Version 3.1 or later for RAMP-TS if using aXes extensions.
- Any newlook software used with RAMP applications must be at version 8.0.5.14769 (or later).
- If you upgrade Newlook software, new license codes may be required.
- The partition to be used must be RDMLX enabled.
- The partition to be used must be enabled for multilingual applications.

Skills

Completed a basic:

- Visual LANSA training course or the equivalent. No advanced Visual LANSA knowledge is required.
- LANSA for the Web course if you are using the Framework for web development.
- WAM tutorial if you want to use WAMs.

Browser Requirements

This version of the Framework software was tested in IE10 and IE11 and the current versions of Chrome, Firefox and Safari.

Using IE10 and IE11 in compatibility view mode is not supported in VLF-WEB.

Only RAMP-TS users can execute IE10 and IE11 in compatibility view mode.

Before Designing and Implementing Any Application!

Review the section [Should You Use the Framework?](#) and the [Application Performance](#) guidelines before designing and implementing any application.

Complete the appropriate [Tutorials](#) in this document before designing and implementing any application.

Warning

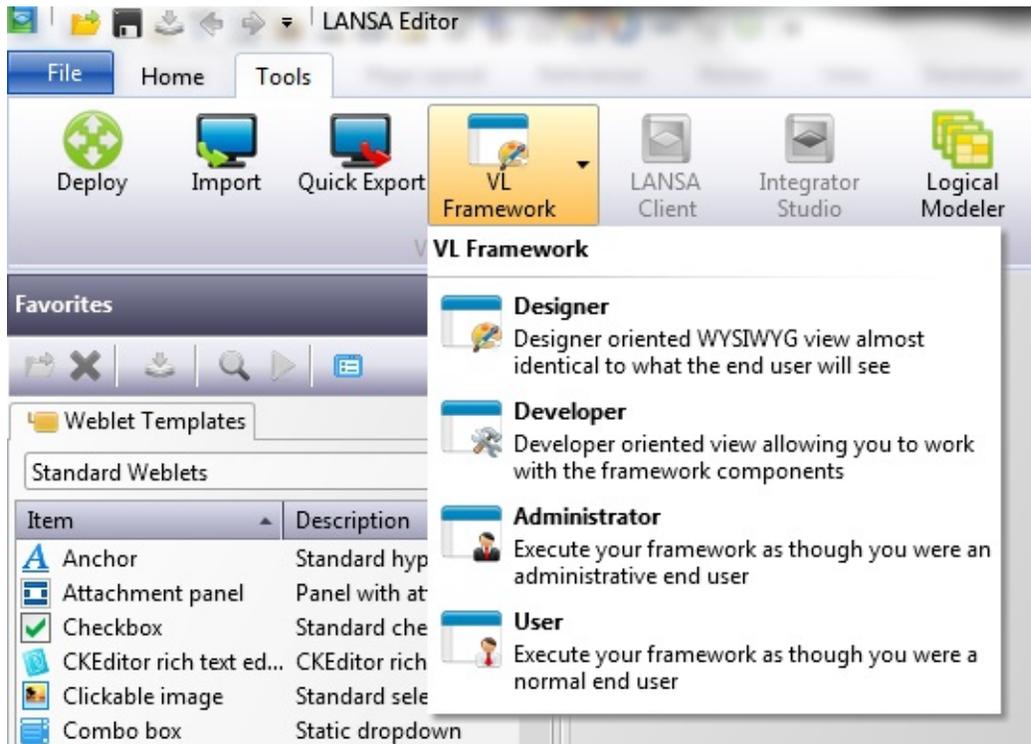
Please do not modify the JavaScript routines shipped with the Framework. If

you do change them, you risk:

- Introducing incompatibilities with future versions of the Framework.
- Voiding or limitation of any maintenance contract you have in place for the Framework.
- Being charged for problem resolutions that are traced back to such modifications.

Starting the Framework

The Framework can be executed in different modes from the Tools ribbon of the LANSa editor:



Designer

In Design mode you can create new applications, business objects, commands etc. and set their properties. You can also work with user and server definitions.

In this mode all filters and command handlers are visible and security settings are disabled. In this mode the performance of the Framework can be slow.

Developer

In Developer mode you gain quick access to the Framework and its objects and properties without having to execute the Framework applications.

To start the Developer's Workbench execute form

UF_DEVEL.

Administrator In Administrator mode you can work with user and server definitions.

User In User mode you can work with the defined applications and business objects but you cannot change them.
Test all your applications in this mode.

You define and prototype applications in the Framework. The snap-in filters, command handlers and other objects for the real application are created using the LANSAs editor.

Should you use Windows or Web Browser Applications?

The Visual LANSA Framework can be used to design and implement applications that use native Windows (Visual LANSA) or Web Browser (LANSA for the Web) user interface technology.

It can also produce applications that are a mixture of both interface technologies.

In deciding whether to use Windows or Web browser user interfaces you need to identify the types of end-users you will have and then to balance their different needs with development and deployment costs:

[Core Users and Occasional Users](#)

[Using a Unified Technology Does Not Mean You Can Have a Single User Interface](#)

[The Zero Deployment Advantage](#)

Core Users and Occasional Users

One way you could do this is to group your users as **Core Users** and **Occasional Users**.

Core Users have one or more of these characteristics:

- Require access to the greatest range of functions that the application provides and use many of them most of the time.
- Use the application as a key component of their jobs (sometimes all day, everyday).
- Are experts at using the application and require interfaces designed for experts.
- Are heavily affected in productivity and morale terms by response times.
- Require a high degree of integration with other applications on their desktops.
- Usually have high-speed TCP/IP connection to the server system typically from inside the corporate firewall.
- Usually work from a fixed location and/or use same workstation (fixed or mobile).
- Are often directly employed or sub-contracted by your organization.

Occasional Users have one or more of these characteristics:

- Require access to a limited range of the functions that the application provides.
- Use very few of the application functions and only use them occasionally.
- Are novices at using the application and require interfaces designed for novices.
- Are not really affected in productivity and morale terms by response times.
- Often only have Internet HTTP connection to the server from outside of the firewall.
- Are often mobile, accessing the application from changing locations and from different workstations.
- May have business relationships with your organization, but are not employed by it.

In user interface terms what this means is:

- Core users usually need fully functioned Windows interfaces (e.g. MS-Excel or MS-Word level interfaces). The interfaces are installed on their workstations or are accessible from high-speed servers. The applications are

designed for maximum functionality, performance and productivity. The high cost of deploying and maintaining core users is counter-balanced by the productivity and performance that they gain by using native Windows interfaces.

Occasional users usually can use a Browser interface. The interfaces are deployed dynamically and are limited to the range of interface functionality provided by technologies such as DHTML. The loss of some of the productivity and performance benefits that a native Windows interface would give to them is balanced by the much lower cost of deploying and maintaining them.

Using a Unified Technology Does Not Mean You Can Have a Single User Interface

Core Users and Occasional Users require fundamentally different user interfaces, regardless of the technology used to deploy the interface to them. Therefore using a single user interface technology does not necessarily mean you have a single user requirement.

There is a very important difference between the interfaces supplied to Core and Occasional Users which is inherent in the way that they use applications: core users require expert interfaces, whereas occasional Users expect interfaces designed "for the less skilled".

For example, imagine that in an accounting application there is a function called "Create Expense Claim".

Expense claims are created by experts in the accounting department and also in self-service mode by general company employees (e.g. Sales staff, engineers, cleaners, etc).

Would you provide the same functional interface to both the user groups? The likely outcome is that the accounting department would complain about poor productivity and the rest of the company would complain that the interface was too hard to understand.

The Zero Deployment Advantage

Web browser applications offer one extremely significant advantage over Windows applications. It is called zero deployment. They can be used anywhere, by anybody, at any time (subject to authority).

To execute a native Windows application you need to deploy it to the users desktop or to a server that the user has LAN speed access to. This process is complex, time consuming, costly and error prone.

However, as a medium for deploying advanced Windows desktop functionality, zero deployment is the traditional Web browser application's only advantage. In all other functional respects, these applications are inferior to native Windows applications. You can make browser applications look and feel like native Windows applications to an extent, but ultimately, and for fundamental architectural, security and bandwidth reasons, they will almost always produce clumsier and slower end-user responses, with poorer navigation and desktop integration capabilities.

It's important to understand that attractiveness of doing everything via a browser interface is largely an economic one.

Zero deployment is a very attractive economic proposition.

In other words, end-users tend to have web browser interfaces imposed on them by IT for cost saving economic reasons rather than for productivity improvement reasons. Often they would much prefer to have the power, speed and flexibility of native Windows interfaces (eg: to the level of MS-Word and MS-Excel) but simple economics and the limits of technology means that they cannot.

You should never think that you are going to actually improve your end-users work life by giving them a browser interface in place of a well designed native Windows one. You are giving them a browser interface to save money.

Getting this balance, between zero deployment economic reality and end-user expectations and requirements, right is a complex equation. If you look on the Internet you can now find reports from various IT industry groups that describe situations where designers and developers have gotten this balance wrong.

The impact of this type of decision can range from end-user complaints, demoralization and loss of productivity through to complete project/product failures. This is especially true in situations where the end-users hold all the decision power (eg: in the purchase of packaged software solutions).

So, should you develop Windows or Web browser applications?

The answer is that in any medium to large-scale commercial application you are most probably going to need to develop both:

- Use Windows interfaces to satisfy the high performance, high functionality and desktop integration requirements of your core users.
- Use Web browser applications to minimize the cost of deployment to your occasional users.

There is, however, one exception to the above rules: [VLF.NET Applications](#). They provide a near-zero deployment capability while maintaining many of the performance, appearance and functional benefits of Windows applications.

Setting Up Your Framework Environment

The Framework can be used to design and implement applications that use native Windows (Visual LANSA) or Web Browser (LANSA for the Web) user interfaces.



You have to [Install the Framework on the Server](#) if you want to:

- Develop any type of Web Browser Application. In this case you need to install the Framework server software and also complete the web server and Framework configuration steps.
- Use RAMP in a development environment, even if your RAMP application is going to be native Windows. In this case you need to install the Framework server software, but you can omit the web server and Framework web configuration steps



You also need to [Install and Configure the Framework on Visual LANSA Workstation\(s\)](#).

Install the Framework on the Server

Follow these installation steps for the type of server you are using:



[System i Apache Web Server](#)



[Windows Web Server](#)

System i Apache Web Server



These steps apply when an Apache web server is run on a System i

Steps	Required for RAMP	Required for Web
Step 1. Ensure that LANSA for the Web is installed and operational on your server		√
Step 2. Make sure that your Apache HTTP server is configured to support server side includes and optionally VLF.NET application deployment		√
Step 3. Ensure that Extended Exchange is enabled		√
Step 4. Install the Framework software	√	√

You may want to refer to [How to start the HTTP server Administration \(ADMIN\) facility](#) and [Can my Web browser applications be used with System i multi-tier web server configurations?](#)

Step 1. Ensure that LANSAs for the Web are installed and operational on your server

Please execute a LANSAs for the Web process or function to confirm the validity of your LANSAs for the Web system. Failure to do this may cause compounding error situations that are difficult to resolve.

Basic troubleshooting:

Ensure the Apache http server instance is started. On the System i, ensure that the Apache http server instance is started. To start it type:

```
STRTCPSVR SERVER(*HTTP) HTTPSVR(<<name of the Apache server instance>>)
```

Ensure the web user profile has appropriate access to the images directory. On the System i, ensure that the profile being used for web access has Read/Write (*RWX) authority to the images directory, (Use WRKLNK to find the directory on the System i, and then option 9 to work with authorities).

Ensure the link between images alias and the actual IFS directory is defined correctly. Always define the link between the images alias and the actual IFS directory using the HTTP server Administration (ADMIN) facility.

You may want to refer to [How to start the HTTP server Administration \(ADMIN\) facility](#).

IBM Web Administration for i

Setup **Manage** Advanced | Related Links

All Servers **HTTP Servers** Application Servers

Running Server: ADMIN - Apache Server area: Include /QIBM/UserData/HTTPAdmin/conf/admin-c

ADMIN > URL Mapping

URL Mapping ?

Aliases **Redirects** URL Rewriting User Directories

URL to host file system mappings: ?

	Alias type	URL path	Host directory or file
Example	Alias	/icons	/QIBMProdData/HTTPAdmin/conf/admin-c
Example	Script Alias	~/cgi-bin(.*)	/www/webserver1/cgi-bin\$1
<input type="radio"/>	Alias	/images	/LANSAIMG/

Add

OK Apply Cancel Preview

Done Trusted sites 100%

Step 2. Make sure that your Apache HTTP server is configured to support server side includes and optionally VLF.NET application deployment

- Start the HTTP server Administration (ADMIN) facility. (See [How to start the HTTP server Administration \(ADMIN\) facility](#)).
- Expand Server Properties and click Dynamic Content and CGI.
- Click the Server Side Includes tab in the form and select the option Allow Server Side Files with CGI program calls inside.
- Add the file extension .pgm (i.e.. a full stop followed by pgm) into the file extensions
- Click OK.

If you are deploying a VLF.NET application see [Deployment](#).

Step 3. Ensure that Extended Exchange is enabled

The Enable Extended Exchange option has to be selected if the Data Application Server is in System i.

Using the LANSAs for the Web Administrator, connect to your server system. From the Tools menu, choose Configure System, then choose Data/Application Server. Select Miscellaneous tab and verify that Enable Extended Exchange is enabled (that is, the checkbox is checked).

Step 4. Install the Framework software

The Framework server software is installed (or updated) by performing a Partition Initialization. You can choose which partition to initialize with the VL Framework during a LANSa install or upgrade or anytime thereafter.

If you choose not to initialize a partition during a LANSa install or upgrade, you can do it using the Work with Administration Tasks option in the LANSa main menu, then choose the VL Framework.

How to start the HTTP server Administration (ADMIN) facility

Start the Admin server:

- On the System i, on a command line, type:

STRTCPSVR SERVER(*HTTP) HTTPSVR(*ADMIN)

Now use the admin server with your browser:

- Enter the URL of your host IP address to start the HTTP server ADMIN facility on the browser:

```
http://<Your Host Address>:2001/HTTPAdmin
```

- Click the Manage tab.
- Select your HTTP Server (powered by Apache) from the Server list.

Windows Web Server



These steps apply to a Windows web server

Step 1. Ensure that LANSAs for the Web is installed and operational on your server.

Step 2. Make sure that your HTTP server is configured to support server side includes

Step 3. Install the Framework software

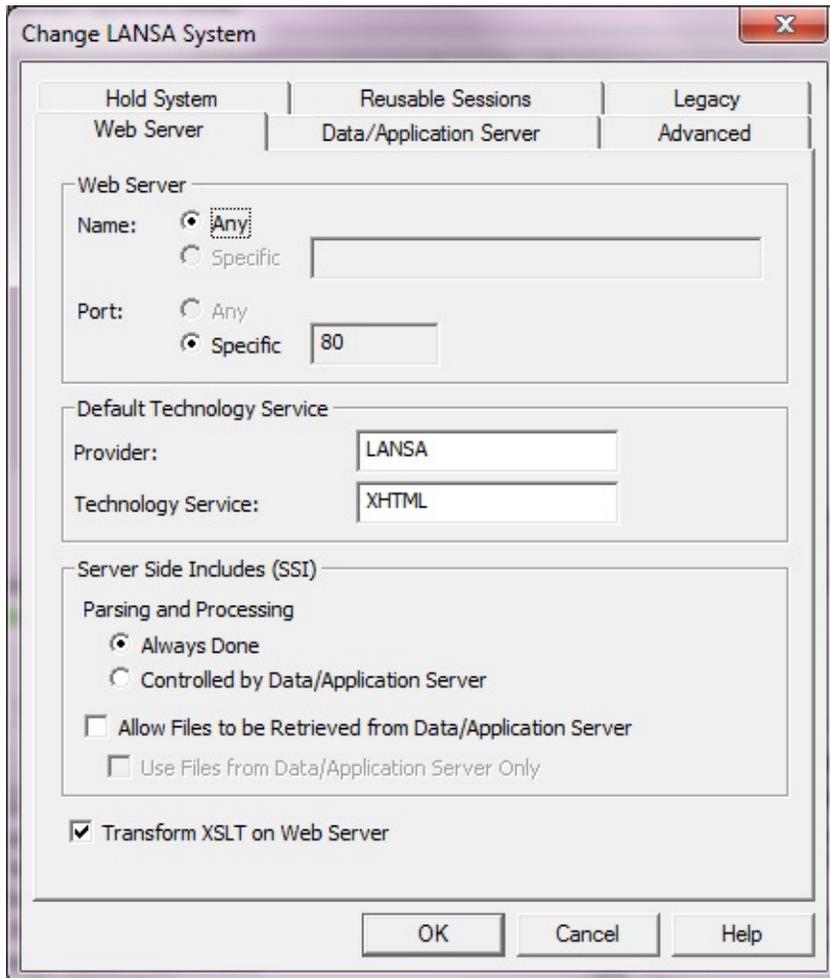
You may want to refer to [Can my Web browser applications be used with Windows multi-tier web server configurations?](#)

Step 1. Ensure that LANSAs for the Web are installed and operational on your server.

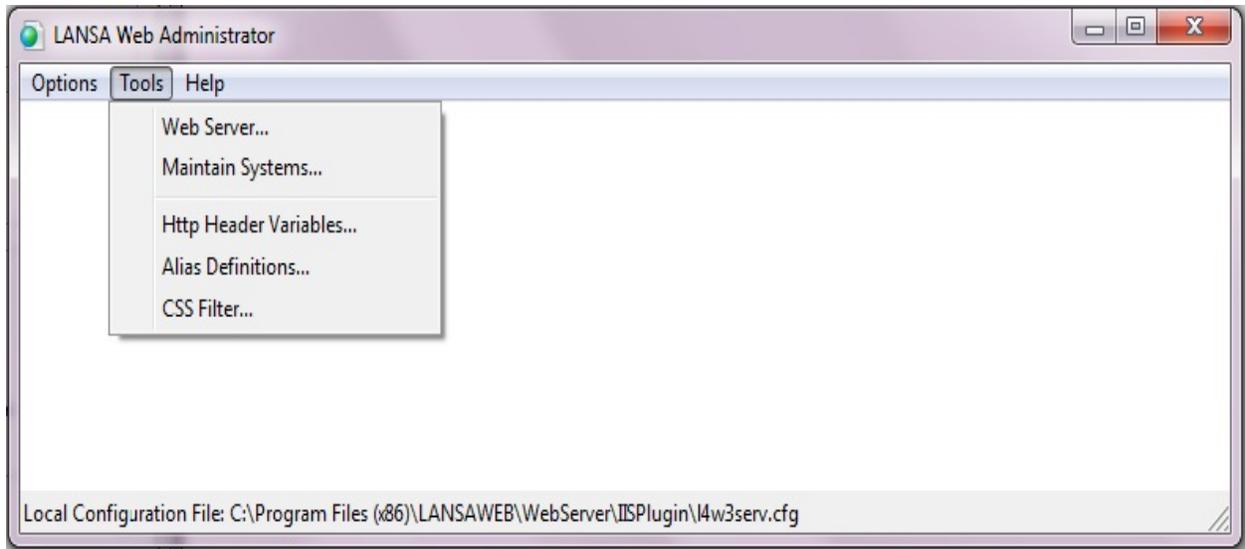
Please execute a LANSAs for the Web process or function to confirm the validity of your LANSAs for the Web system. Failure to do this may cause compounding error situations that are difficult to resolve.

Step 2. Make sure that your HTTP server is configured to support server side includes

Use the Web Administrator to make sure your LANSAs System configuration has the Server Side Include (SSI) support enabled:



You get to this dialog by opening your local LANSAs web configuration file and selecting Tools/Maintain Systems:



When changing Web Administrator settings remember to save the Configuration and always restart Web Server in IIS.

Step 3. Install the Framework software

- Start the Visual LANSa IDE and log on to your chosen partition.
- Check whether Visual LANSa Framework options are on displayed on the Tools tab on the ribbon.
- If the Visual LANSa Framework is not installed log off and log on again.
- On the Visual LANSa IDE logon screen click on the Partition Initialization button and select the Visual LANSa Framework option.
- Wait until the installation completes before proceeding to the next step.

To view these applications you must use the WAMS=N start up option. For more details please see [Web Application Start Options](#).

Install and Configure the Framework on Visual LANSAs Workstation(s)



Every Visual LANSAs workstation you are using for development needs to have the Framework software installed on it.

[Step 1. Install the Base Visual LANSAs Framework Software](#)

[Step 2. Configure the Visual LANSAs Workstation\(s\)](#)

Step 1. Install the Base Visual LANSAs Framework Software

Warning: If you already have the Visual LANSAs Framework installed these steps will restore your Framework back to its “as shipped” state, removing any hot fixes you may have applied.

- Start the Visual LANSAs IDE and log on to your chosen partition.
- Check whether Visual LANSAs Framework options are displayed on the Tools tab on the ribbon.
- If the Visual LANSAs Framework is not installed log off and log on again.
- On the Visual LANSAs IDE logon screen click on the Partition Initialization button and select the Visual LANSAs Framework option.
- Wait until the installation completes before proceeding to the next step.

Note: Before starting ensure the partition you are going to use is RDMLX and multilingual enabled.

Step 2. Configure the Visual LANSA Workstation(s)

Each Visual LANSA workstation you are using needs to have the Framework software configured to link to your LANSA for the Web development system.

[Step 1. Verify use of Microsoft Internet Explorer](#)

[Step 2. Enable the Framework for Web Browser applications](#)

[Step 3. Set up your Windows webserver or your iSeries webserver](#)

[Step 4. Save your Framework details to your web server](#)

[Step 5. Test execution of your Framework in a Web Browser](#)

Note: If the "Do you want to save your Framework" message box pops up while you are specifying the following details, reply NO. You will save your Framework changes towards the end of the following steps.

Step 1. Verify use of Microsoft Internet Explorer

Start Internet Explorer and use the Help then About Internet Explorer menu options.

Check that the version shown is 10.0 (or later).

Otherwise install a required version of Internet Explorer before proceeding.

Step 2. Enable the Framework for Web Browser applications

Start your Visual LANSA development environment using your chosen partition. Then use the Tools -> VL Framework - As Designer option to start the shipped Framework as a designer.

Use the (Framework) and then (Properties...) menu items to display the Framework properties folder. Switch to the Framework Details tab.



Check:

- Enable Framework for Web Browser Applications.
- Enable Framework for WAMs if you intend to create LANSA for the WEB WAMs.
- Enable Framework for AJAX style applications
- Compile Framework as Microsoft .NET executable if you intend to create VLF.NET applications.

Switch to the User Administration Settings tab and **uncheck** the Use Framework Authority and Users check box. (For the moment we are disabling the user and authority checking to simplify the installation and verification process. Later on these options can be enabled again, if required.)



Shut down the Framework and save the changes you have just made.

Step 3. Set up your Windows webserver or your iSeries webserver

Prerequisite Steps

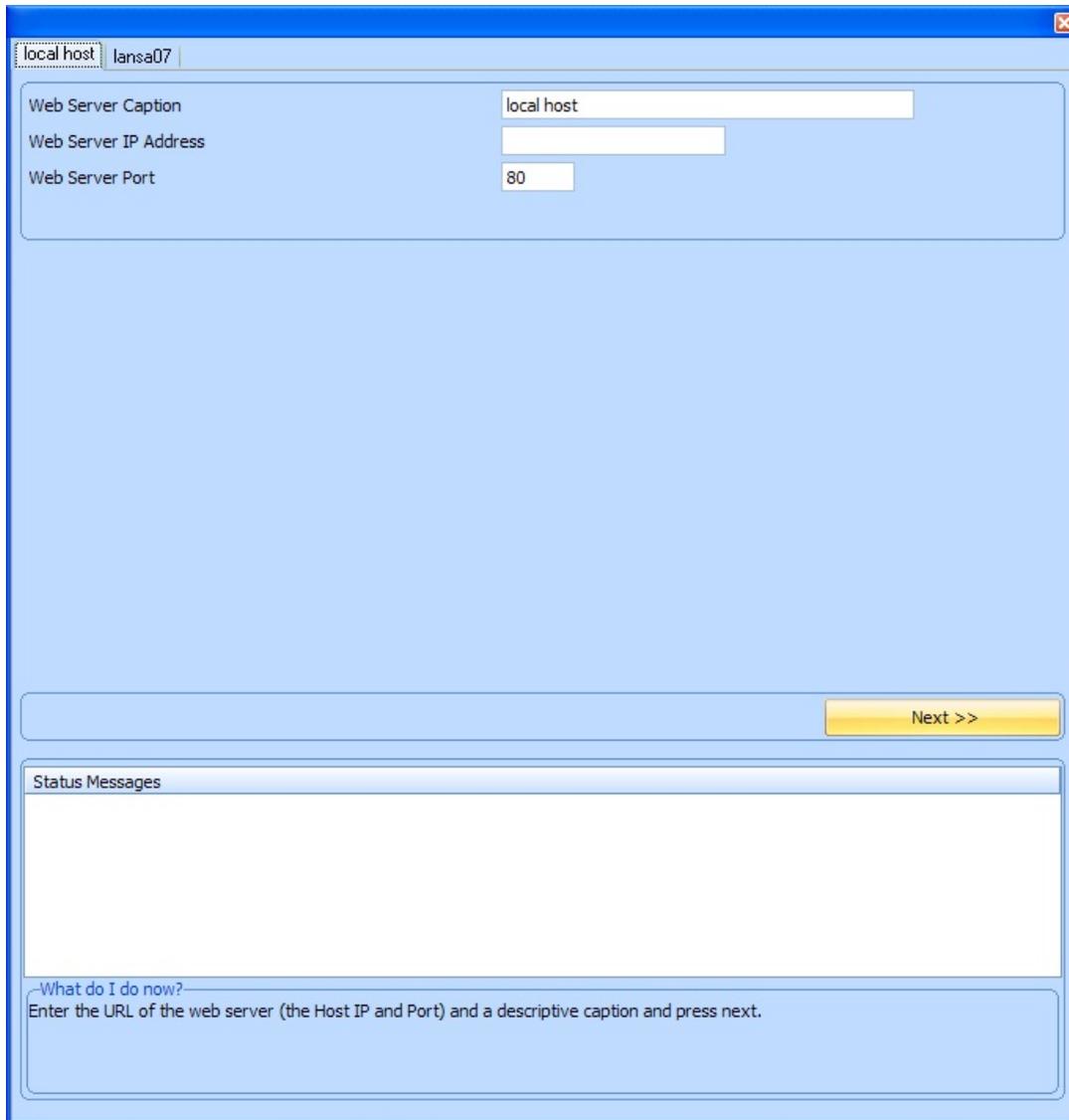
Before you start, it will make configuration easier if you ensure that the following has been done:

1. LANSA for the Web is installed, configured and started on the web server.
2. The partition you are working with on the web server is enabled for LANSA for the Web, and has been initialised.
3. The partition you are working with on the web server is enabled for RDMLX.
4. The VLF (epc870 or later) has been imported into that partition on the web server.
5. The web server is accessible from your PC.
6. If the webserver is located on a different machine to your development machine, a network drive has been mapped to the web server and the drive letter used has been recorded.

Run the Web Configuration Assistant

Start the Web Configuration Assistant from the VLF (Framework) --> (Web configuration assistant...) . For more information see [Web Configuration Assistant](#).

Two tabs are displayed. Each tab can be used to define a web server to the Framework:



local host | lansao7

Web Server Caption: local host

Web Server IP Address:

Web Server Port: 80

Next >>

Status Messages

What do I do now?
Enter the URL of the web server (the Host IP and Port) and a descriptive caption and press next.

Windows Web Server

In one of the tabs, enter the ip address and port used by your windows web server, and a caption describing the webserver.

For example, if the webserver was an IIS webserver on your own PC, and you were using port 80, you could enter:

My Windows Webserver | lansa07

Web Server Caption: My Windows Webserver

Web Server IP Address: localhost

Web Server Port: 80

Check using URL - http://localhost:80

Press the Next button and follow the instructions.

The Web Configuration Assistant will attempt to automatically determine everything it can about your web server, but sometimes it will need to request some information from you:

- It may need to know the location that `http://localhost/Images` points to
- It will always need to know what **Private Working Folder** you want to use (if the folder does not exist, the assistant will create it). Enter the value and press next, then Save, then move on to the next tab:

My Windows Webserver | lansa07

Web Server Caption: My Windows Webserver

Web Server IP Address: localhost

Web Server Port: 8080

Check using URL - http://localhost:8080/cgi-bin/lansaweb?procfun+VF_PR004+VFU0411+EX1+FUNCPARMS+VF_WKAct(A0010):" +...

The type of server: WINNT

The webserver's images URL path: /images

The location of the images directory on the server: C:\PROGRAM FILES\LANSAWEB\WEBSERVER\IMAGES\

Private Working Folder: my project

Temporary Folder Name: vif_temporary_files

Next >>

Status Messages

- ✓ A webserver has been found at this IP and Port
- ✓ LANSAs for the web is enabled for this lansa installation
- ✓ LANSAs for the web is enabled for this lansa partition and the VLF is installed
- ✓ The directory on the server associated with /images was successfully detected.

What do I do now?
Enter a Private working folder name

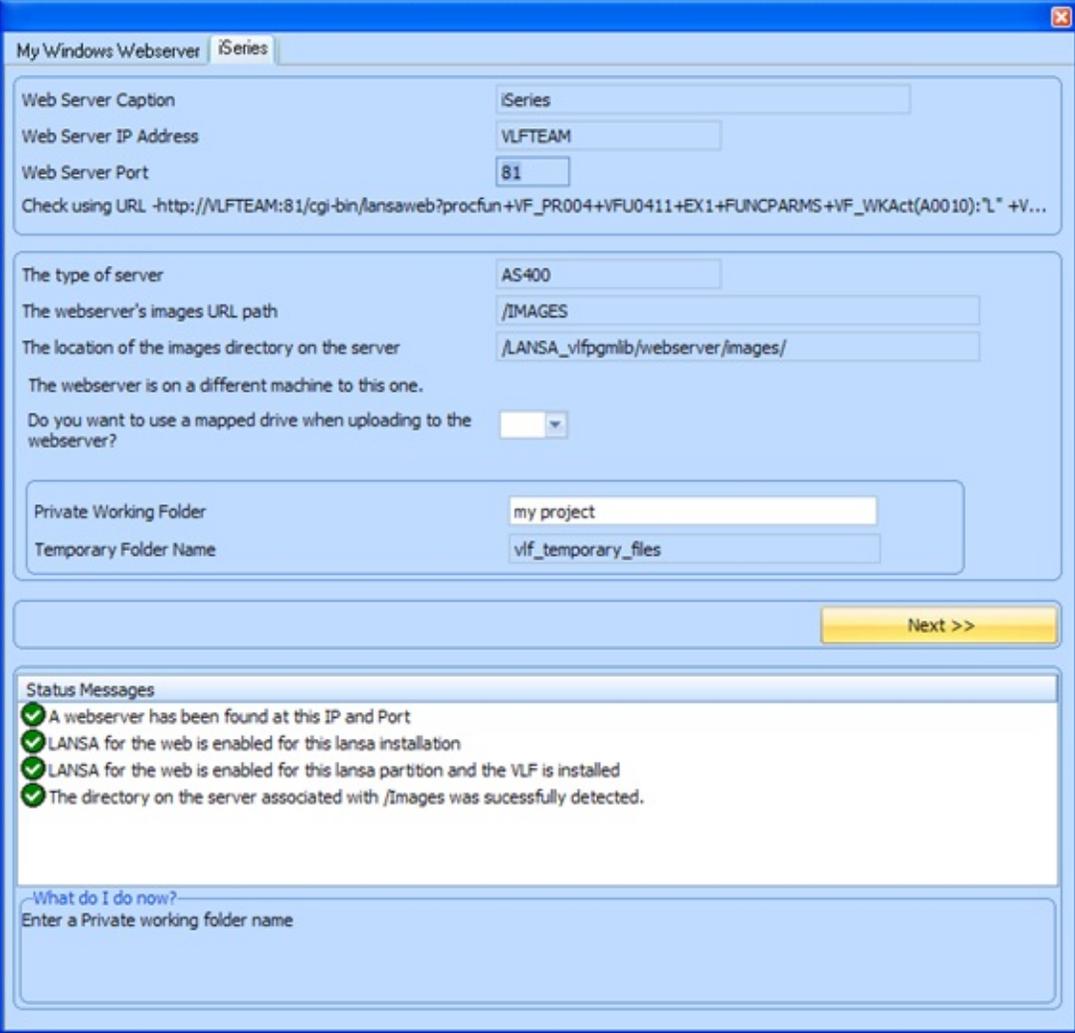
iSeries web server

Configure the iSeries web server in the same way.

The main difference is that it will be necessary to specify a mapped drive if you want the VLF to upload your web framework to the web server automatically.

To map a drive to an iSeries that is on your network, go to windows explorer, and choose the option Tools --> Map a network drive.

Choose a drive letter and map it to `\\host_ip\IFS`, and map it using a different user name - use an iSeries user profile and password (where host_ip is the ip address of the iSeries, or a name that resolves to that ip address).



The screenshot shows the 'My Windows Webserver' configuration window for an iSeries. The configuration is as follows:

- Web Server Caption: iSeries
- Web Server IP Address: VLFTEAM
- Web Server Port: 81
- Check using URL: `-http://VLFTEAM:81/cgi-bin/lansaweb?procfun+VF_PR004+VFU0411+EX1+FUNCPARMS+VF_WKAct(A0010):"L" +V...`
- The type of server: AS400
- The webserver's images URL path: /IMAGES
- The location of the images directory on the server: `/LANSA_vlfpplib/webserver/images/`
- The webserver is on a different machine to this one. (checked)
- Do you want to use a mapped drive when uploading to the webserver? (unchecked)
- Private Working Folder: my project
- Temporary Folder Name: vlf_temporary_files

A yellow 'Next >>' button is visible at the bottom right of the configuration section.

Status Messages

- ✓ A webserver has been found at this IP and Port
- ✓ LANSAs for the web is enabled for this lansa installation
- ✓ LANSAs for the web is enabled for this lansa partition and the VLF is installed
- ✓ The directory on the server associated with /images was successfully detected.

What do I do now?
Enter a Private working folder name

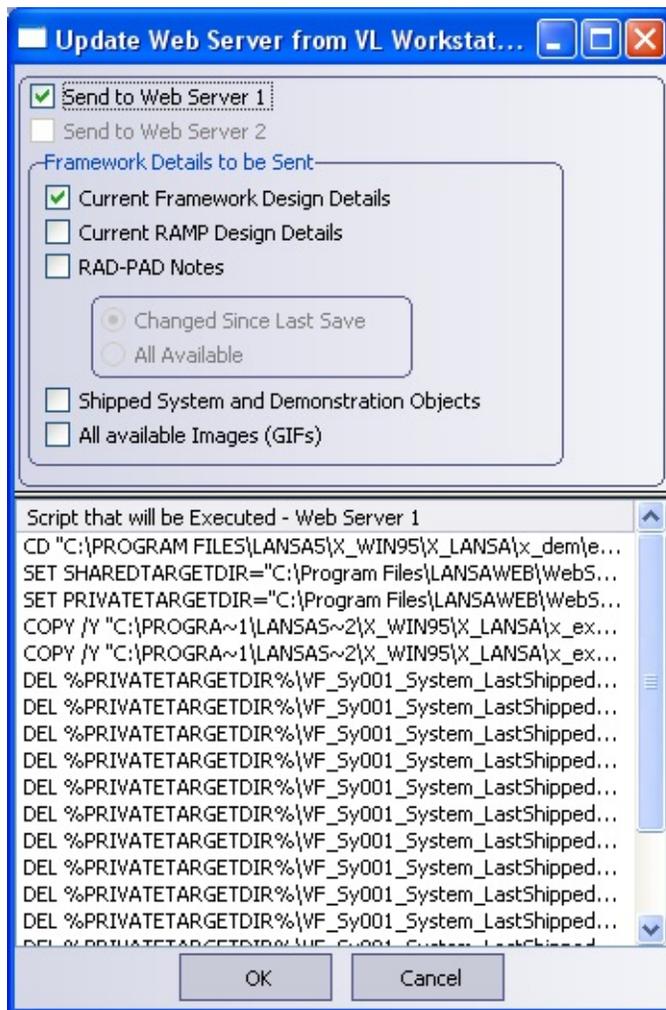
When you have finished, save and then close the Web Configuration Assistant.

Step 4. Save your Framework details to your web server

Save the Framework to save your changes.

Since you now have a LANSA for the Web private working folder, an Update Server from VL Workstation window appears.

It will appear every time you save your Framework from now on:



This window is indicating that it is going to copy your Framework design (and some other things) to your images and private working folders on your web server(s).

Since this is the very first time you have done this since installing or updating your web server you should:

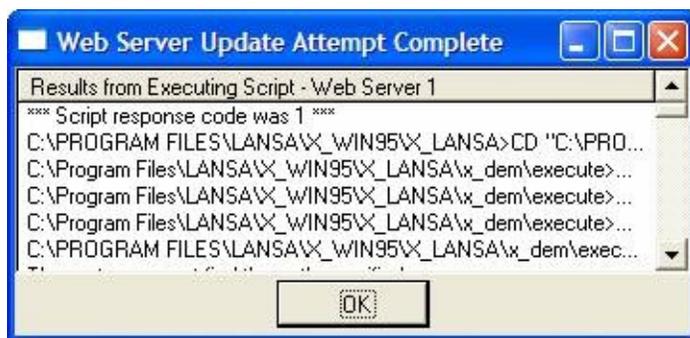
- Uncheck the **RAD-PAD Notes** option.
- Check the Shipped System and Demonstration System Objects option.
- Check the All available Images (GIFs) option.

Click OK and wait until the copy operations complete.

Note 1: Since you are copying everything to the server this operation may take several minutes to complete. Subsequent copies to your server will have much less data and thus take a lot less time. Normally you only copy everything after you have installed the Framework or performed an upgrade to it.

Note 2: Generally mapped drives are faster than FTP, so if you have a mapped drive available you should use it.

When the copy operation completes, the update form will reappear stating "Results from Executing Scrip - xxxxx" (where xxxxx is the caption you assigned to your web server) like this example:



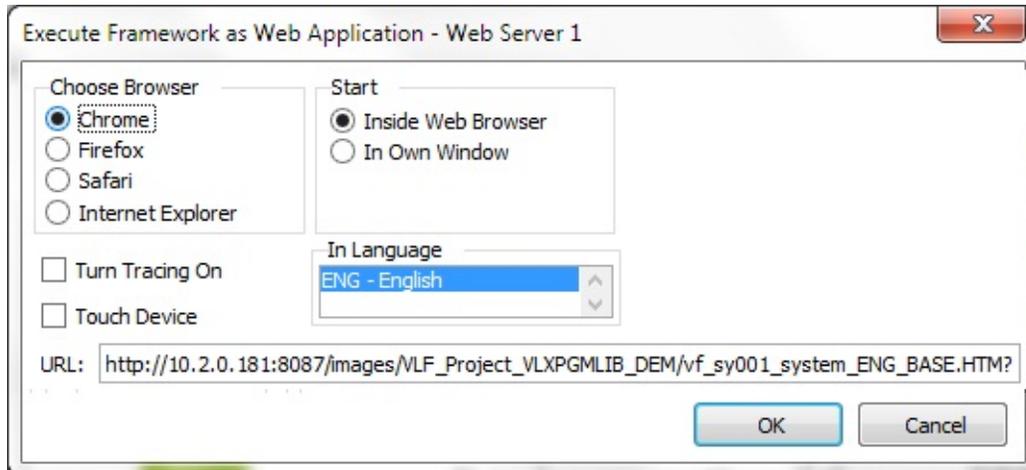
Scroll through the script messages looking for indications of failed COPY or FTP operations.

Click the OK button to close the update form.

If an FTP or COPY operation fails it is most probably because the IP and/or directory details you supplied on the associated [Developer Preferences](#) tab are incorrect. Return to the tab for the web server and carefully check all supplied details (use F2 to get additional information about each option). Use the various Verify options to check for problems. When you have corrected the cause of the problem recommence step 4.

Step 5. Test execution of your Framework in a Web Browser

Use the (Framework) -> (Execute as a Web Application...) menu options to execute your Framework as a Web browser application:



Uncheck the Turn Tracing On option and click OK. Your Framework should start inside a browser.

The first time you use the Framework as a Web browser application it may be slow to start up as the various files it uses will not be in your web browser's file cache.

Do the [Tutorials](#) to quickly learn how to design or implement Web browser applications.

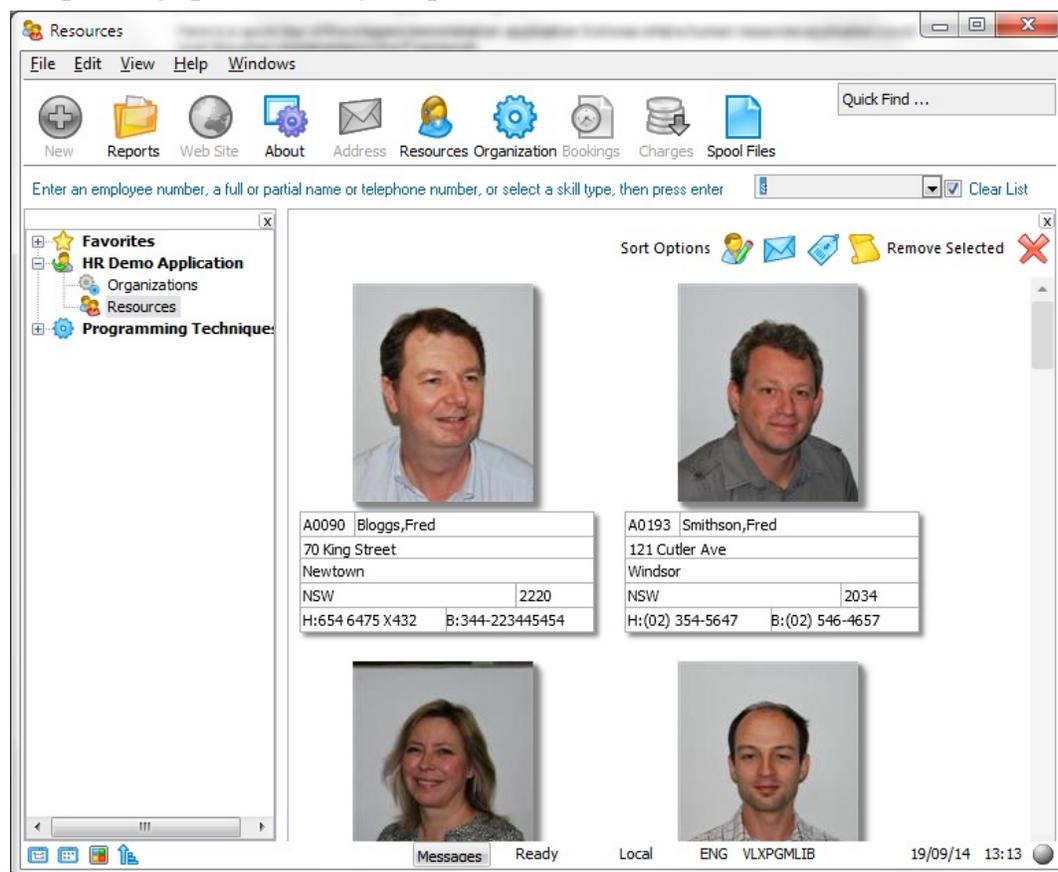
The Demonstration Application

Important: The shipped demonstration components should be executed using Render Type M (Mixed).

Here is a quick tour of the shipped demonstration application. It shows what a human resources application could look like when implemented in the Framework.

Its aim is to highlight how you can use the Framework to:

- Add value to the user experience – via navigational structures, ease of use/access, etc.
- Add value to the business – via integration (desktop and application), charting, reporting, productivity improvements, etc.



The various business objects and commands it contains use quite different styles. They are designed to demonstrate some of the many techniques you can use with the Visual LANSA Framework product:

- [HTML Startup Page](#)
- [Graphical Content](#)

- [Complex Grids](#)
- [Web Content](#)
- [Google Maps Gadget Integration](#)
- [Wizards](#)
- [Data Entry](#)
- [Dynamic Report Generation](#)
- [Integration of desktop and folders into a VL application](#)
- [Integration with Microsoft Excel](#)
- [Video Content](#)
- [Mini-Filters](#)
- [Generic Notes Command Handler](#)
- [Generic Spooled Files Command Handler](#)
- [IBM i Server Message Queues](#)

Later on you may want to look at the code of the demonstration application. The parts of the demonstration application are shipped in components starting with the prefix DF_*

The demo is HR system for a mythical software engineering company ACME. It uses these LANSA demonstration database tables: PSLMST, PSLSKL, PSLTIMES, DEPTAB, SECTAB and SKLTAB. The application itself presents two main views of the information contained in these tables. One view is by the structure of the Organizations that define the ACME company, and the other is by the Resources that work for ACME.

Initializing the Demo Database Tables

If required, LANSA contains programs to initialize the shipped tables PSLMST, PSLSKL, PSLTIMES, DEPTAB, SECTAB and SKLTAB. To use it execute shipped process PSLUTL. Use the menu option Install Demonstration Data.

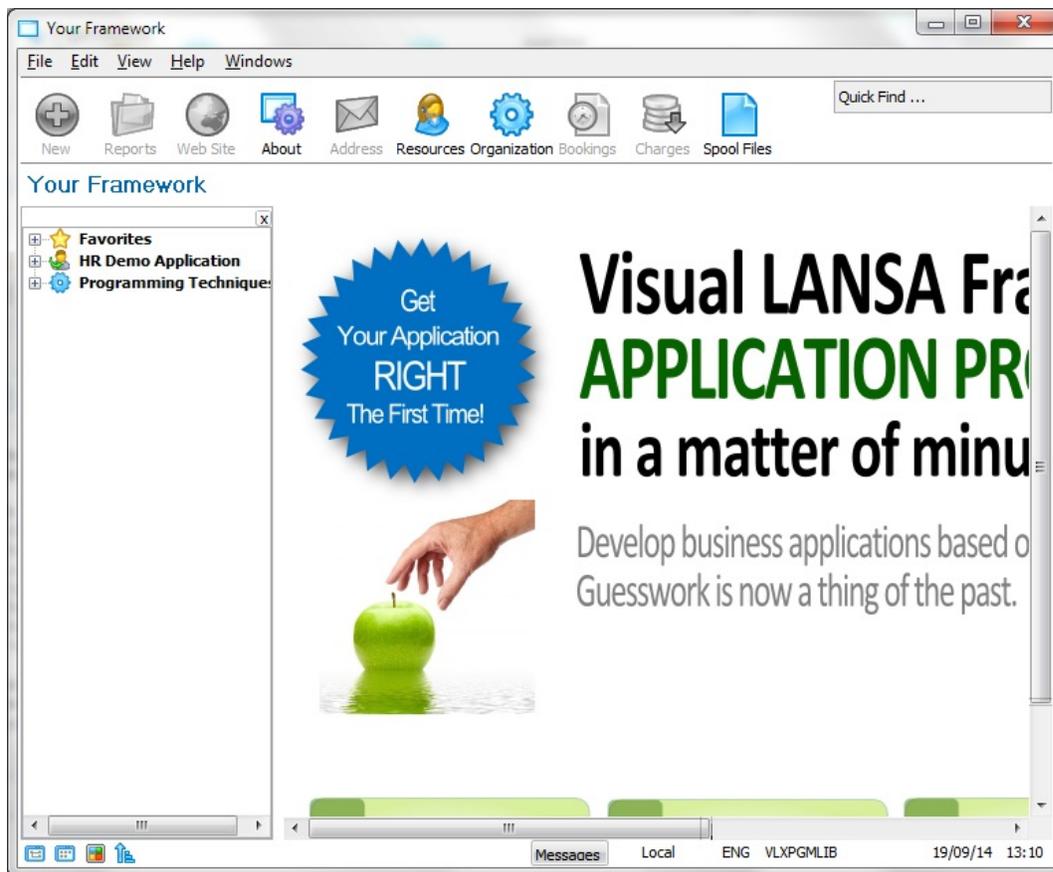
Showing Graphs using the Microsoft Office web controls

To see the graphs on the organization and section details tabs you need to have Microsoft Office 2003 (or later) on the PC and also the Microsoft Office 2003 (or later) web controls.

HTML Startup Page

The start up page is typically used to bind together corporate resources.

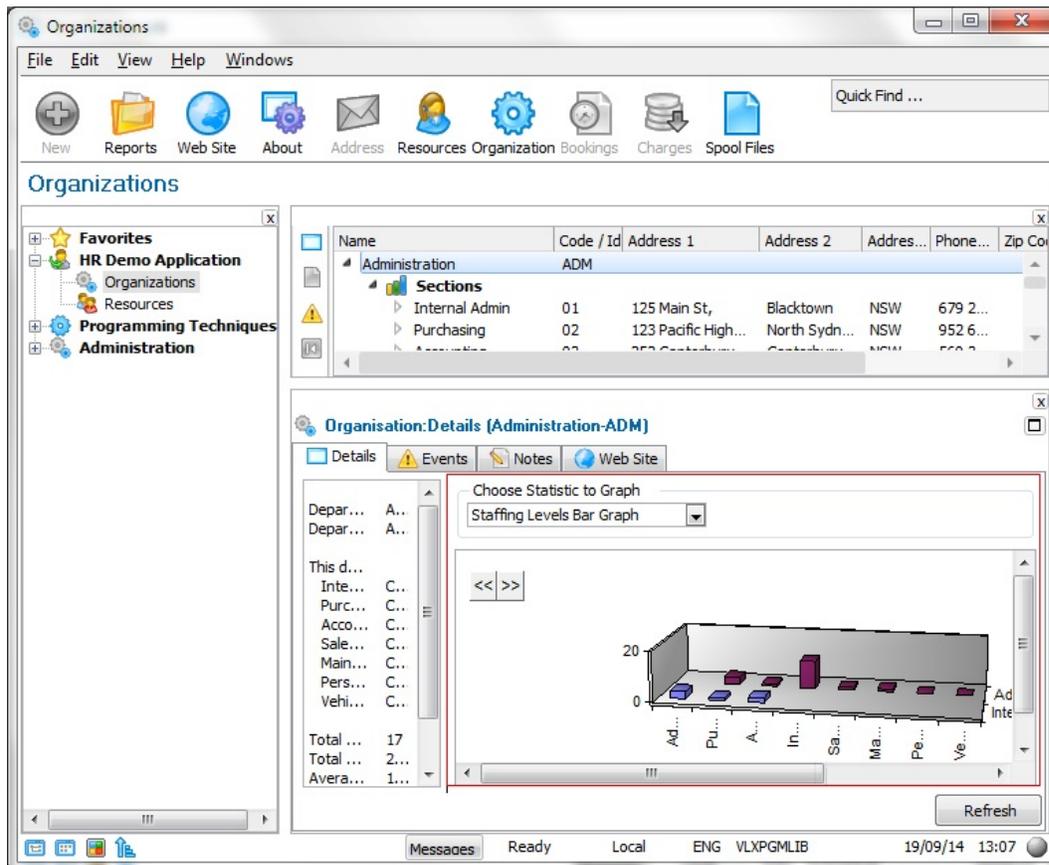
It can be personalized to the corporation's own style and it would provide links to other resources.



Use the special value `*AUTO*` in the [URL](#) property to display an HTML page at start up.

Graphical Content

Organizations is a classic commercial business object with a hierarchical instance list of departments, sections and staff resources with command handlers using graphs:



The instance list tool bar changes according to the type of business object selected. The individual tabs can be shrunk and expanded as required (see [Allow Panes to be Shrunk and Expanded](#)).

Chart layouts can be changed and details appear when clicking on charts.

Visual LANSA has its own easy to use graphing facility, but in this version of the demo application we have chosen to show graphs using an embedded Microsoft Office 2003 web control.

To see the charts you need to have Microsoft Office 2003 on the PC and also the Microsoft Office 2003 web controls.:

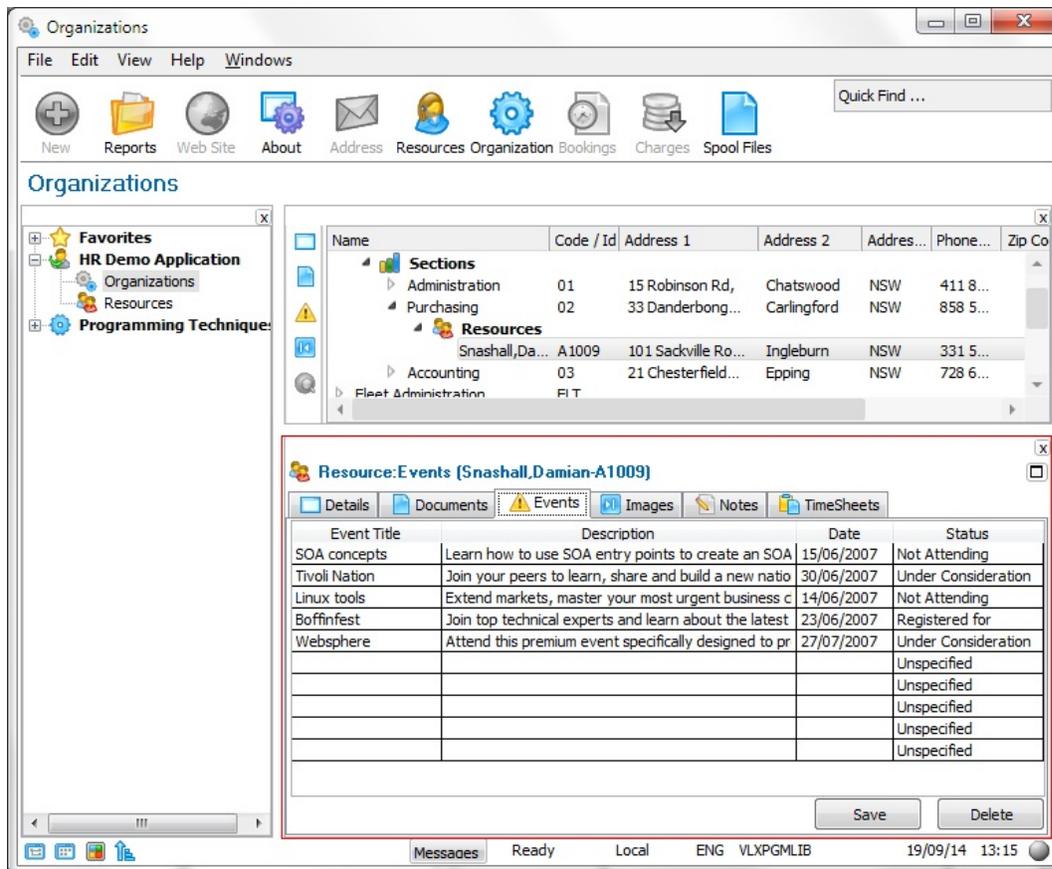
<http://www.microsoft.com/downloads/details.aspx?FamilyID=7287252C-402E->

[4F72-97A5-E0FD290D4B76&displaylang=en](#)

The advantage of this method is that almost the same logic can be used to set up the chart for Windows as for the web.

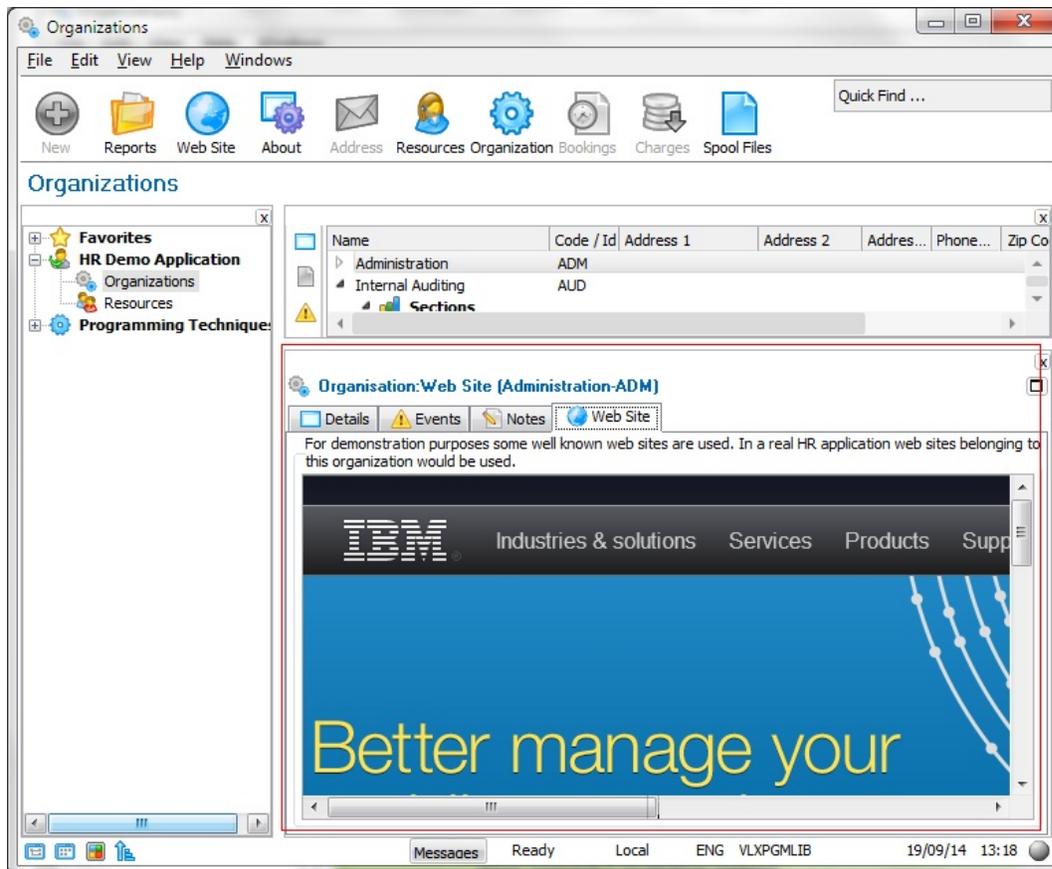
Complex Grids

The grid showing events uses drop-downs and calendars to make editing easy:



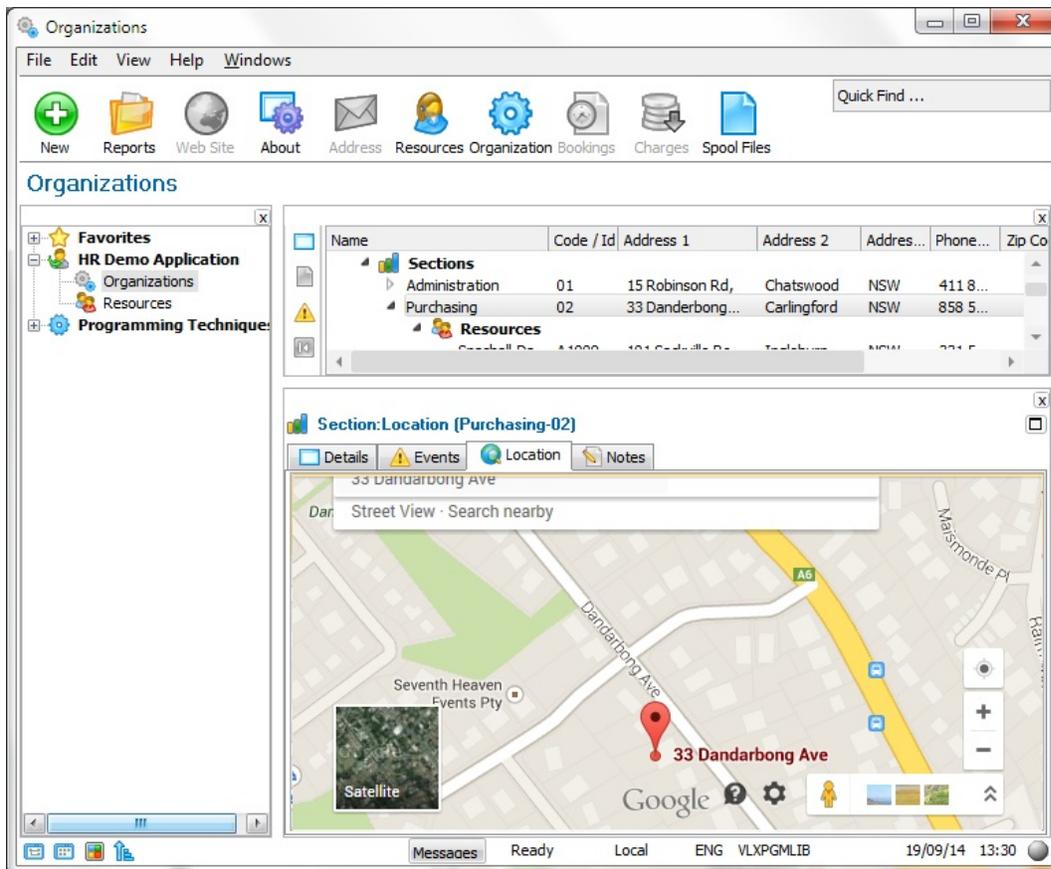
Web Content

Web content can be integrated in the application:



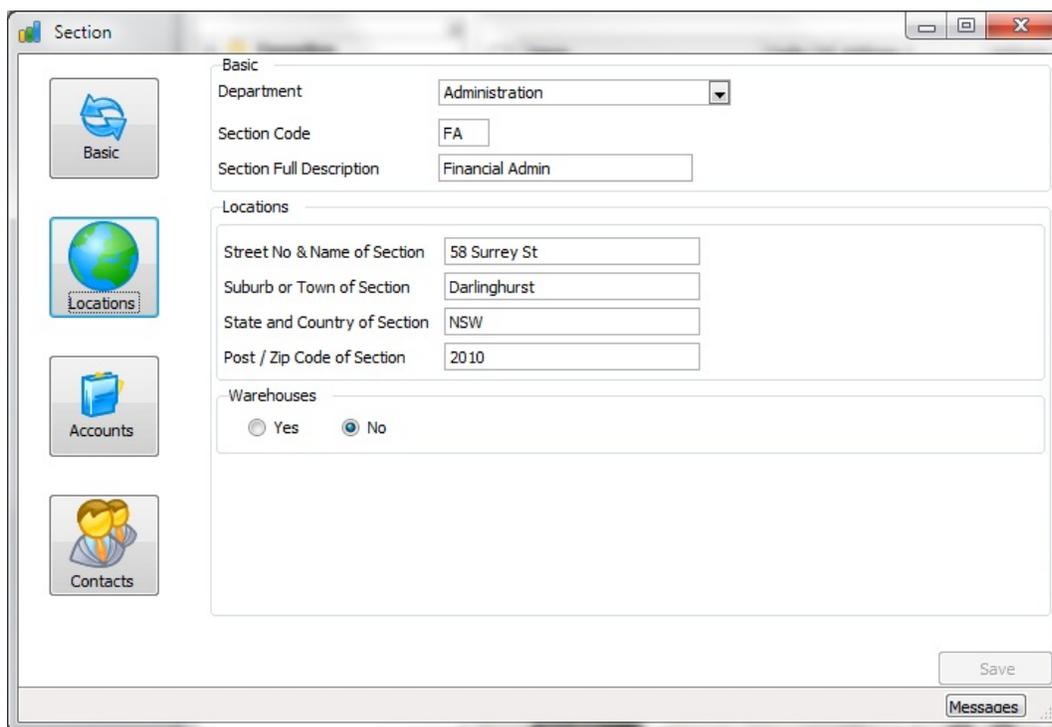
Google Maps Gadget Integration

The Google Maps Gadget is used to display department and section locations:



Wizards

The New option for a section demonstrates a wizard style form used to create a new section. The user is progressed through multiple panels and can move backwards and forwards through them as desired.



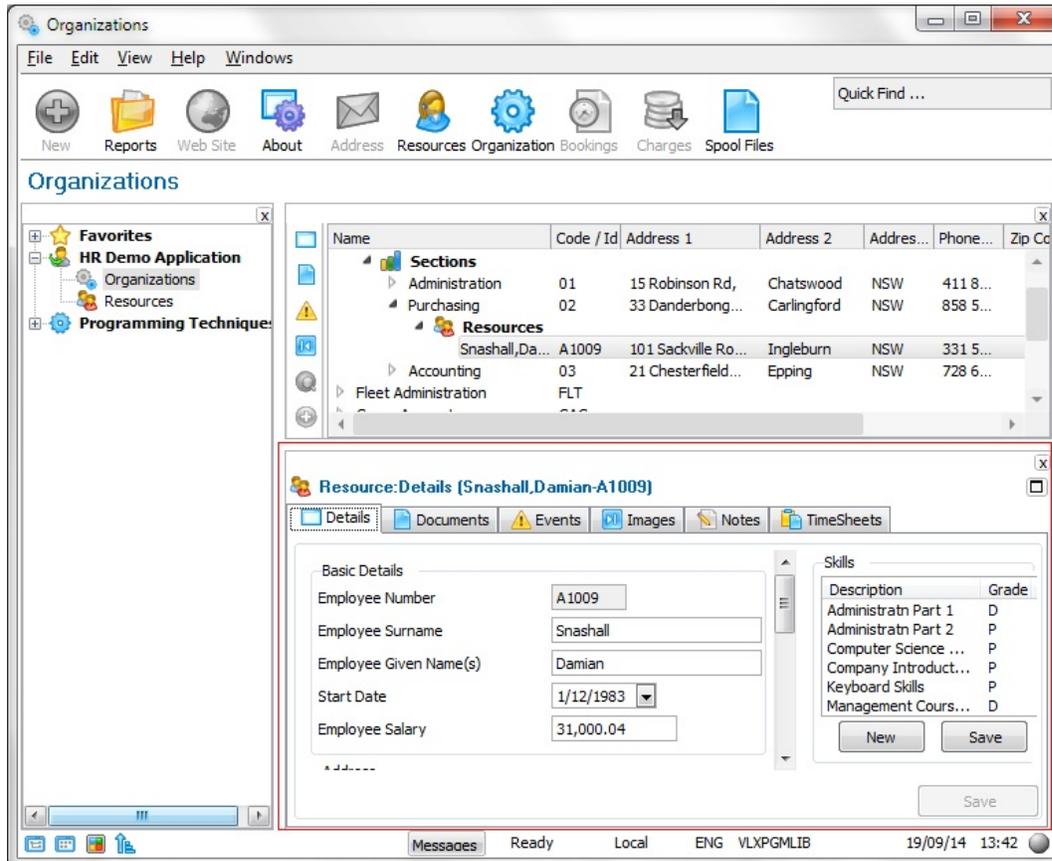
The screenshot shows a window titled "Section" with a sidebar on the left containing four navigation buttons: "Basic" (selected), "Locations", "Accounts", and "Contacts". The main content area is divided into three sections:

- Basic:** Department (Administration), Section Code (FA), Section Full Description (Financial Admin).
- Locations:** Street No & Name of Section (58 Surrey St), Suburb or Town of Section (Darlinghurst), State and Country of Section (NSW), Post / Zip Code of Section (2010).
- Warehouses:** Radio buttons for Yes and No, with "No" selected.

At the bottom right, there is a "Save" button and a "Messages" icon.

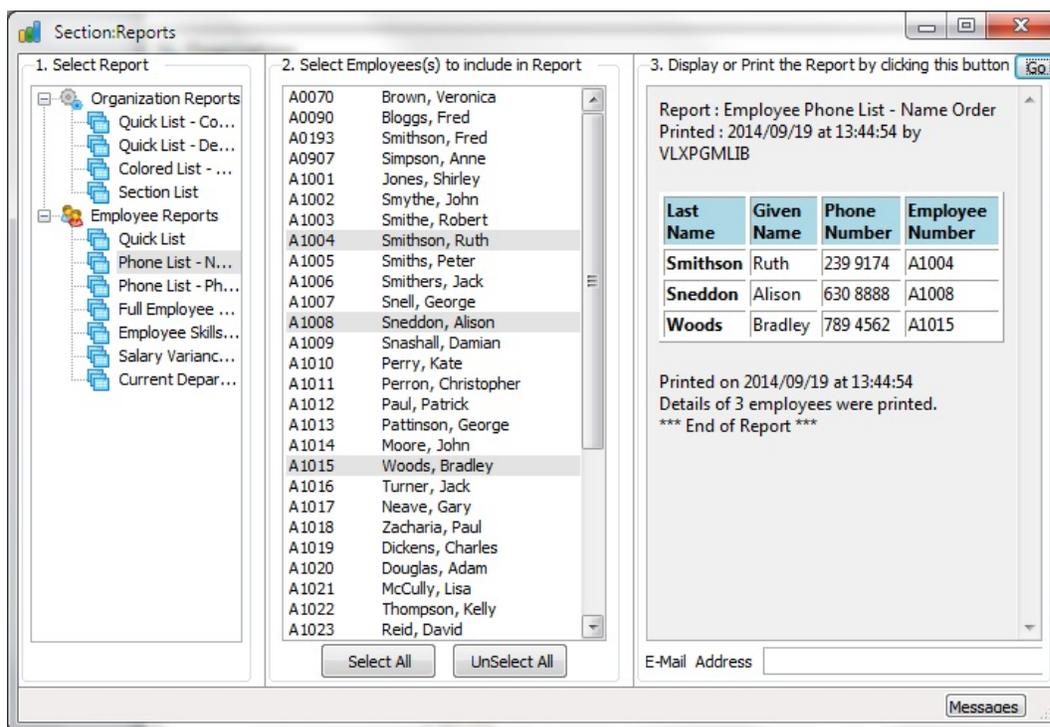
Data Entry

The employee details tab is an example of dense data entry screen. The LANSA Repository rules cause error messages if an entry is incorrect and unsaved changes are trapped.



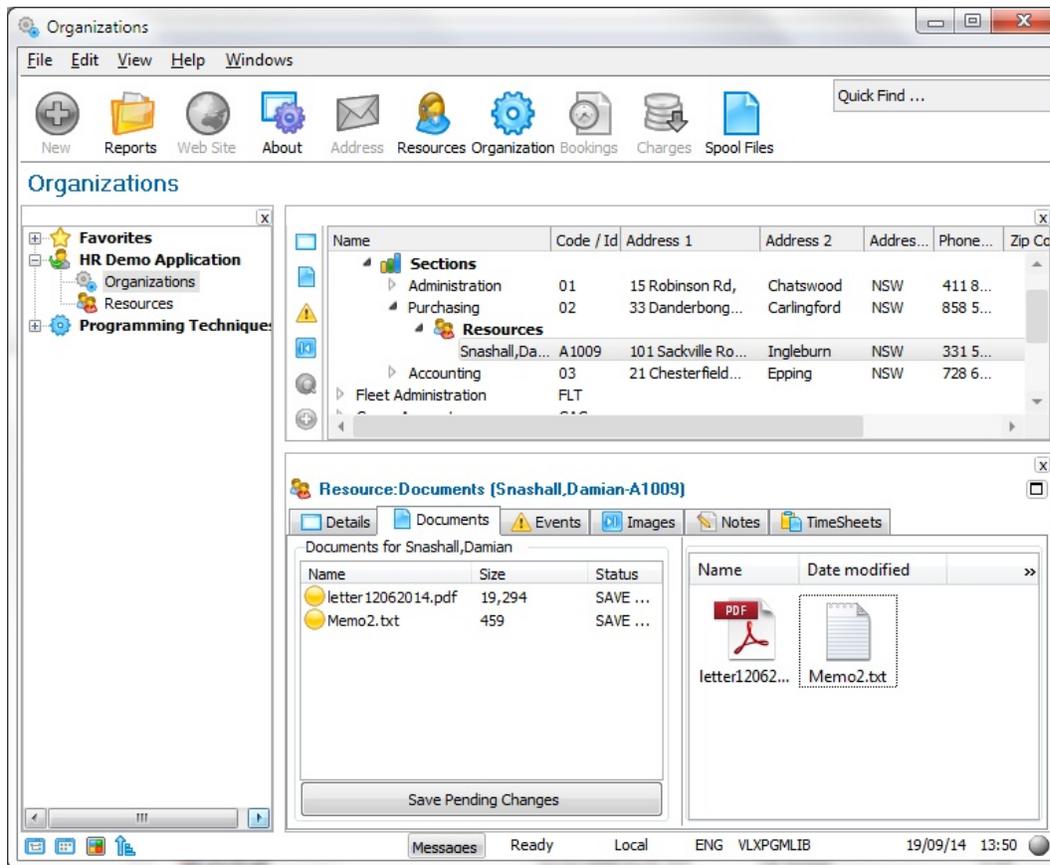
Dynamic Report Generation

The Reports command handler demonstrates dynamic production of reports at user's desktop using integration of reports with MS-Word and MS-Excel and email.



Integration of desktop and folders into a VL application

The employee documents are stored in a database when saved. You can store any type of pc file (for example .pdf, .htm or .gif):



Integration with Microsoft Excel

Excel is integrated in the application:

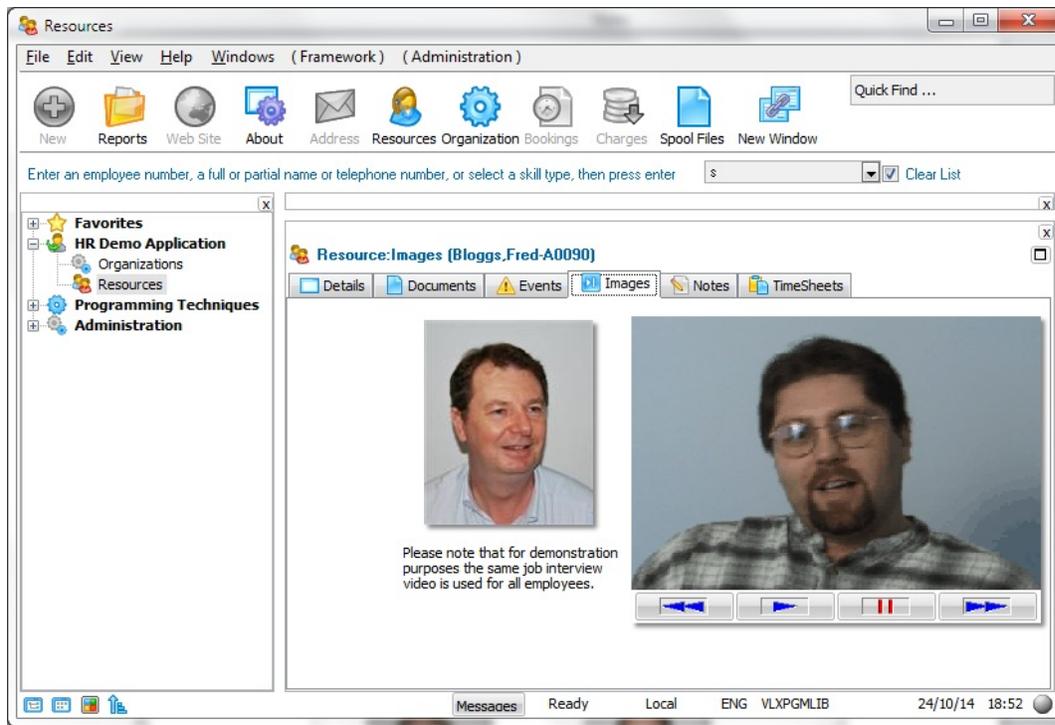
The screenshot shows the 'Organizations' application window. The main content area displays the 'Resource: TimeSheets (Snashall, Damian-A1009)' view. On the left, a tree view shows the hierarchy: First Quarter, Second Quarter, Week 14-26, Third Quarter, Fourth Quarter, 2005, and First Quarter. On the right, a profile card for Damian Snashall shows his photo, name, date started (011283), and monthly salary (2,583.34). Below this, a table titled '2004 - Quarterly Summary' is displayed, which is integrated with Microsoft Excel. The table shows the following data:

Task	Q1	Q2	Q3	Q4
Sick Lea	32.0		8.0	8.0
Public H	24.0			16.0
Totals	528.0	520.0	520.0	520.0

The application interface includes a menu bar (File, Edit, View, Help, Windows), a toolbar with icons for New, Reports, Web Site, About, Address, Resources, Organization, Bookings, Charges, and Spool Files, and a status bar at the bottom showing 'Messages', 'Ready', 'Local', 'ENG', 'VLXPGMLIB', and the date/time '19/09/14 13:56'.

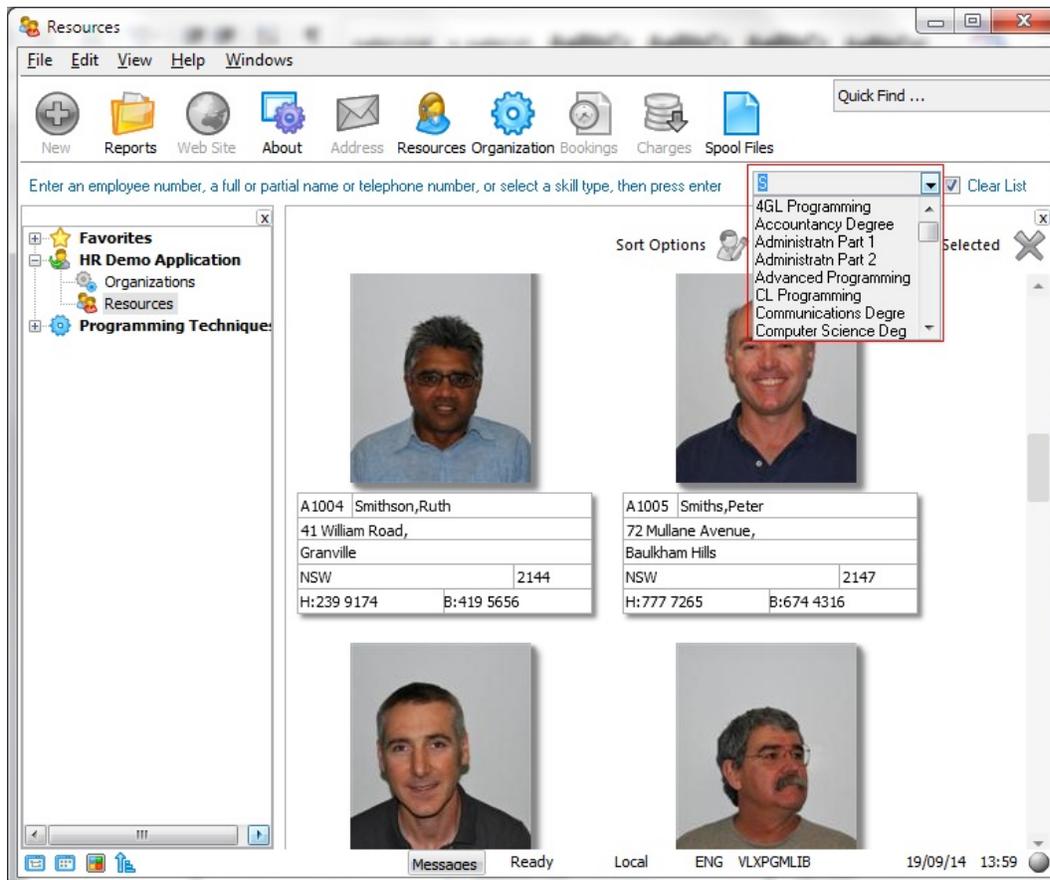
Video Content

Photos and videos are inserted in the command handler:



Mini-Filters

The Resources business object uses a mini-filter which you can use to locate employees in various ways

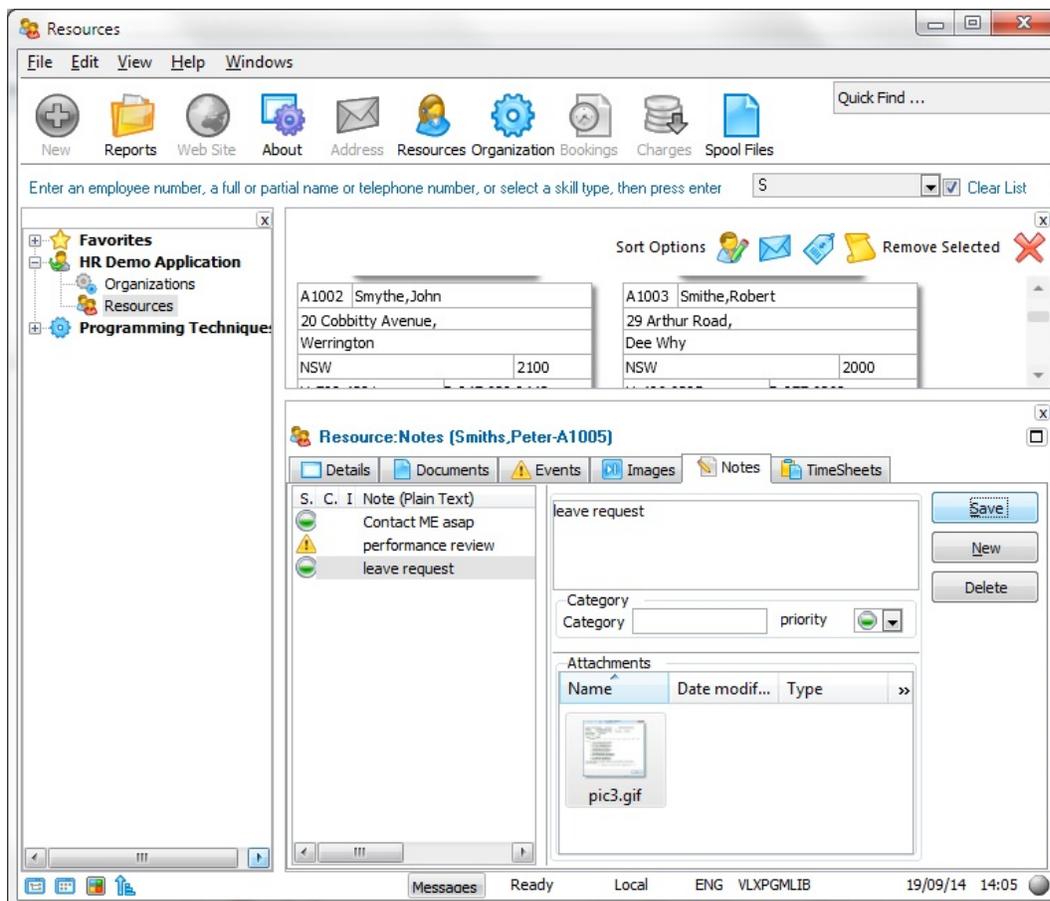


For example:

- Type in employee number A0090 and press Enter.
- Type in name "SM" and press Enter (the search is by scanning, not generic).
- Select a skill from the drop down.
- Uncheck the Clear List option to build aggregate lists.

Generic Notes Command Handler

In the shipped HR Demo Application the Employee Notes tab allows many individual notes to be associated with an Employee.



The Notes command handler DF_T3201 is generically designed and is immediately useable with any business object defined in a VLF framework. For example a Product, Customer or Order business object in a RAMPed 5250 application could have a Notes/Documents/Attachments capability added to it instantly.

Each individual note can be classified, and may have multiple documents/attachments associated with it (eg: MS-Word documents, e-mails, images/photos, etc, etc). All the note and document/attachment details are permanently stored in database tables on the database server, making them instantly available everywhere to all authorized users, without needing additional Windows servers, mapped drives, shared folders, security systems etc.

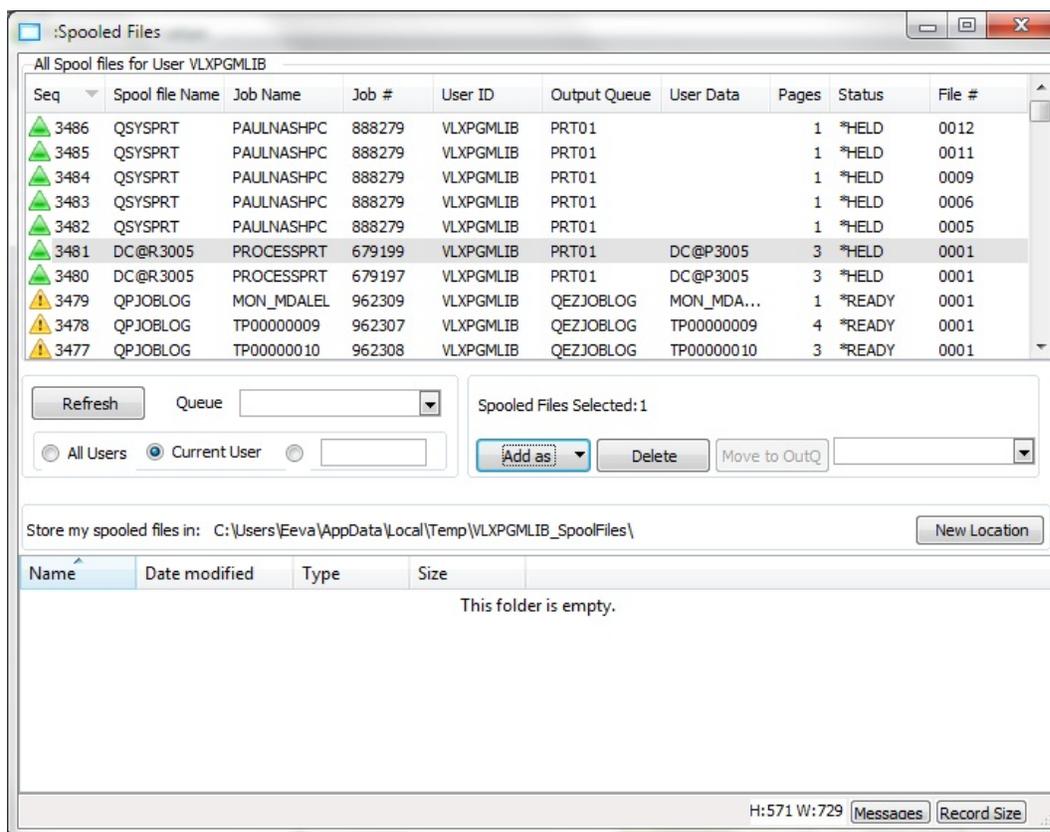
Reusing the Notes command handler this way may add significant value to any 5250 application that is being RAMPed.

See the RAMP tutorial [Snapping in Shipped Events Command Handler](#) for step-by-step instructions.

All the source code is provided and customers are encouraged to copy and modify logic as they see fit.

Generic Spooled Files Command Handler

In the shipped Demonstration System the Spool Files command handler allows the end-users to browse, delete, or change the queue of a spooled file. They can also convert the spooled file to a text file or PDF, which can be copied, emailed, viewed etc..



The spooled files browser example for Windows uses the generic shipped command handler DF_T3101 which can be attached to any framework or business object.

If using the spooled file convert to PDF feature, and if the iSeries does not use CCSID 37, it may be necessary to edit and recompile the spooled file PDF converter program.

See the shipped source for this program for details of the change and how to recompile it. The source is on the iSeries, in file UF_3GSRC, and the source member is UF_3GSPLRP.

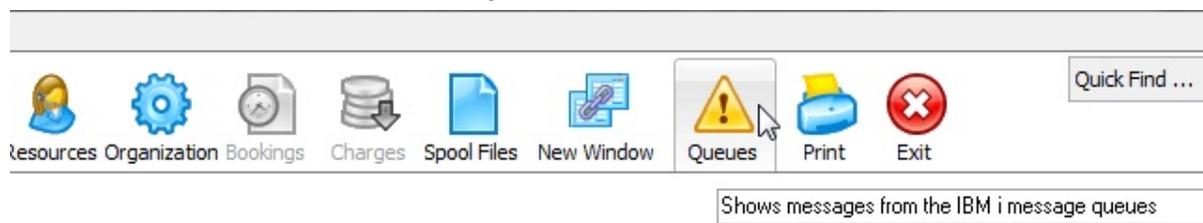
Note that source code member(s) involved may be replaced to their "as shipped" state if you re-install or upgrade the VLF. You need to put in place a process to

check for and reapply your changes are reinstalling or upgrading the VLF.

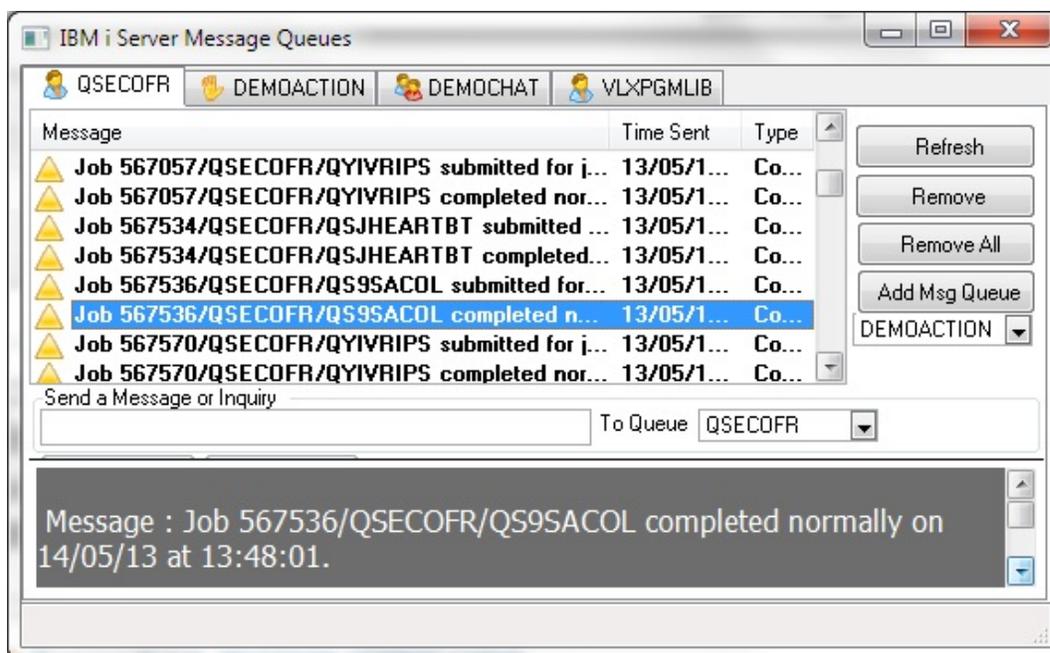
IBM i Server Message Queues

The IBM i Server Message Queues demonstration application is invoked by the Queues button on the tool bar.

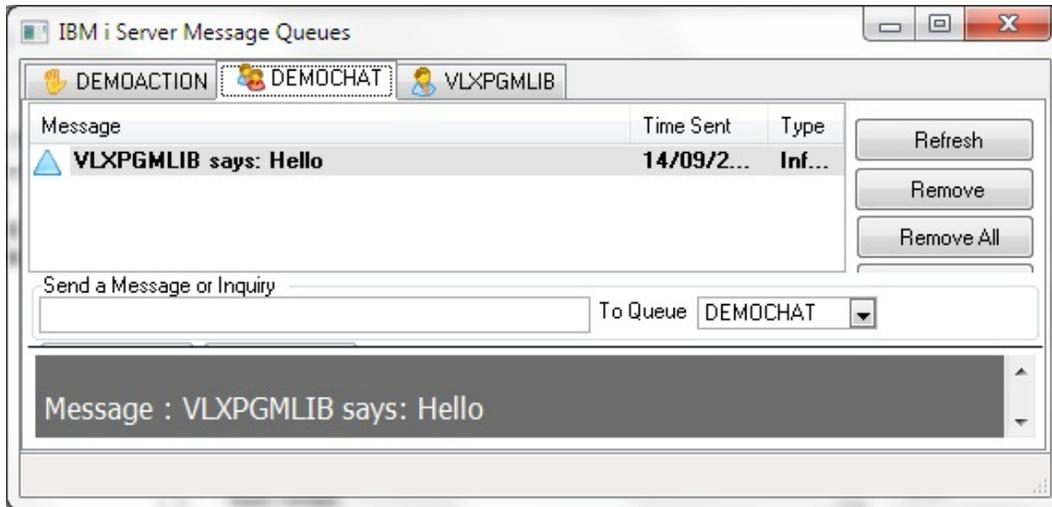
It shows how you can use personal user queues, group chat queues and event notification queues. You need to have a super-server connection to an IBM server active to use it effectively.



Personal queues are just classic IBM i user profile queues – typically used to notify individuals about things like batch job completion, system status, etc.



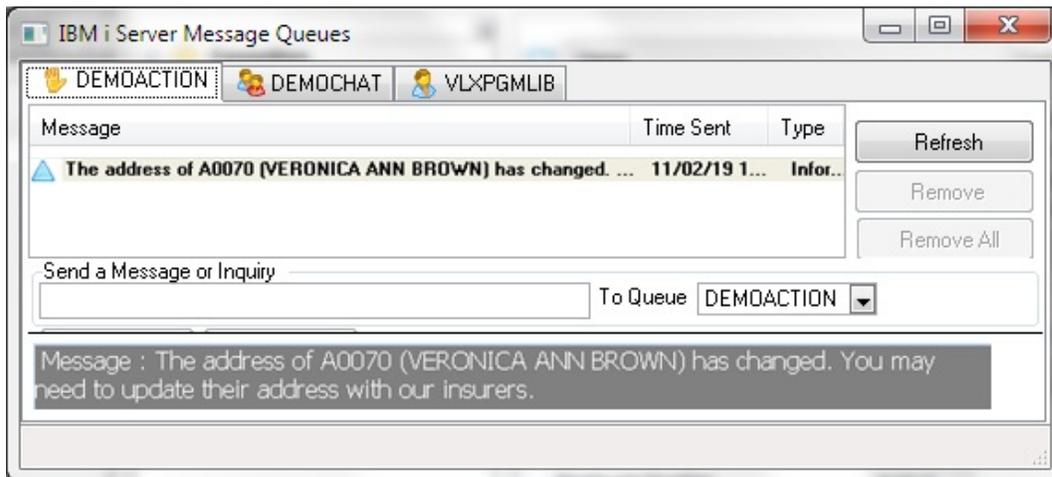
The chat queue example shows how IBM i message queues might allow a group on individuals involved in the same work group to exchange information:



Event notification queues demonstrate a technique for programmatically sending “actionable” messages to groups or individuals via IBM i message queue(s).

When a user decides to “action” this type of message, the VLF responds programmatically by doing things like opening a URL, switching to a business object instance, executing a command handler, etc.

This technique can be demonstrated by updating the address or phone number of an employee in the shipped demonstration HR application – which will cause an “actionable” message to be issued:

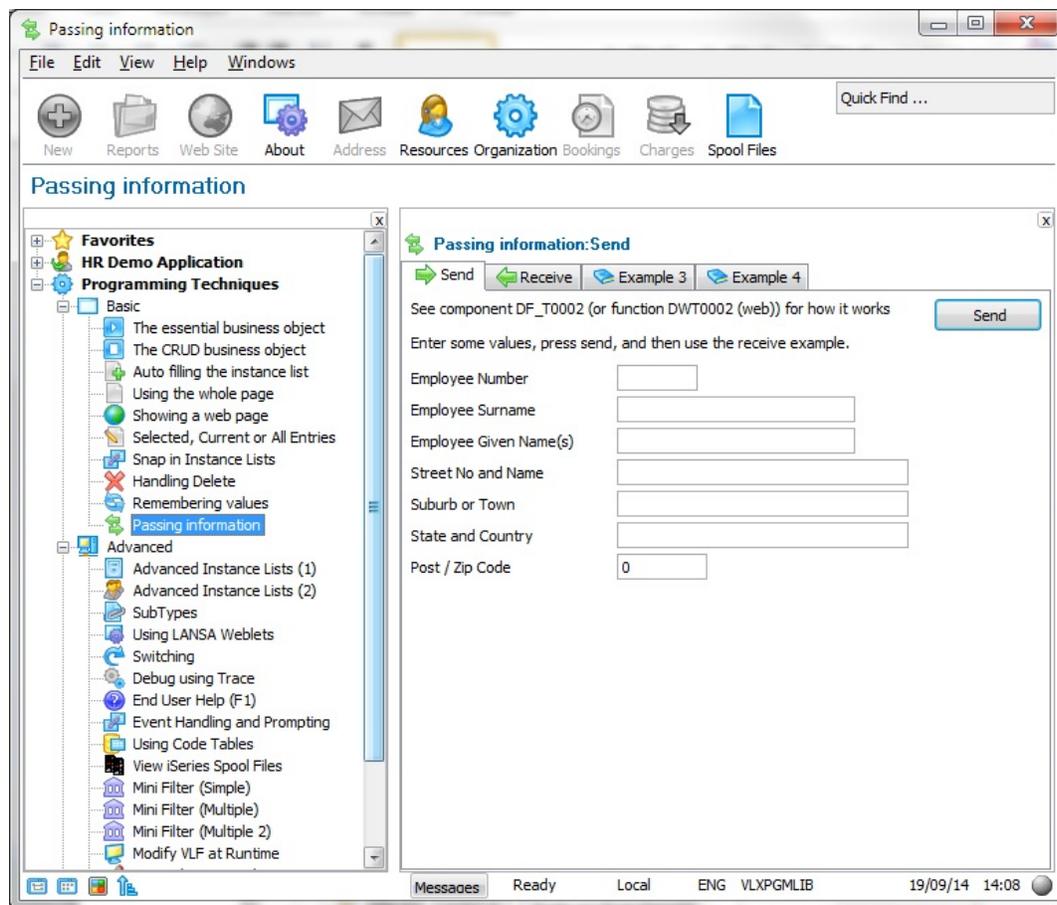


All source code is shipped with the examples.

The Programming Techniques Application

When it comes to producing programs to execute within the Framework the easiest way to get started is to have a look at the shipped Programming Techniques application.

While the shipped Demonstration application shows you what a real application might look like, the Programming Techniques application shows you how to program applications. All the examples provided are shipped with source code, so not only can you execute the examples, you can also examine the source code that drives them. The Programming Techniques application is focused on Web Browser application.



Development Architecture

This section describes how Framework development can be structured and managed:

[Usage Scenario: One Designer – Multiple Developers](#)

[Usage Scenario: Multiple Designers – Multiple Developers](#)

[How to Create a New Framework Version](#)

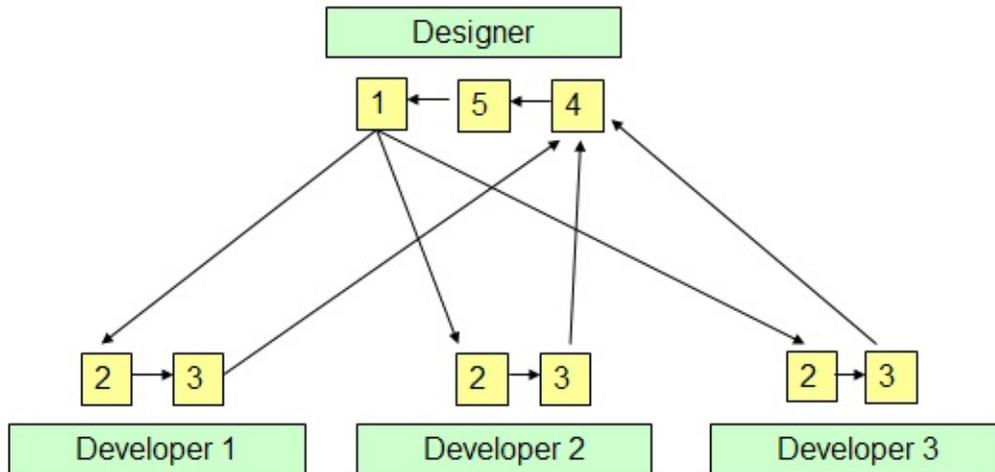
[How to Keep Automatic Backup Copies of Your Framework](#)

[How to Export Framework Definitions](#)

[How to Merge Items from one Framework to Another](#)

Usage Scenario: One Designer – Multiple Developers

This type of scenario is suitable for projects that have a single designer utilizing multiple developers to implement the application:



Step Description

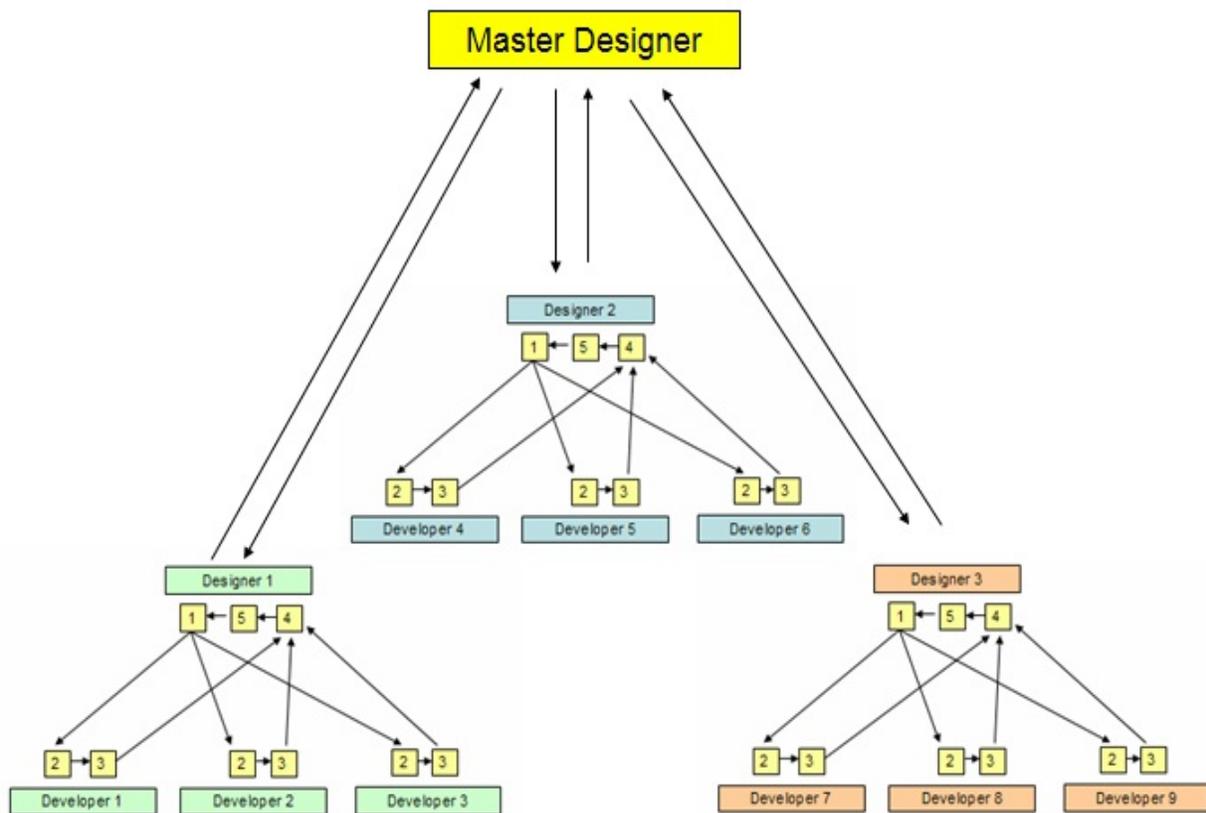
- 1 The designer creates a prototype Framework defining all required applications, views, business objects, filters and command handlers for this development cycle. A copy of the Framework definition XML file is given to developers 1 -> 3.
- 2 Developers 1 through 3 now code, compile and test the various filters and command handlers assigned to them by the designer. To do this they need to snap them into the Framework definition. Note that the developers are not creating new Framework objects. They are modifying the definitions of existing objects only.
- 3 As developers 1 -> 3 complete the filter and command handler work assigned to them they use the Merge Tool to create a merge list containing the filters and command handlers they have altered. The merge lists are e-mailed back to the designer.
- 4 The designer receives the merge lists and merges the changed filter and command handler definitions into the master Framework version.
- 5 Each developer is given a fresh copy of the master Framework definition which contains their completed work and the completed work of the other developers. This new copy of the master Framework becomes the base on

which they will do further work.

This type of development cycle is repeated until all filters and command handlers have been completed. The master Framework is gradually evolved from a prototype into a completed application ready for deployment.

Usage Scenario: Multiple Designers – Multiple Developers

Sometimes large projects require the use of multiple designers and teams. These can be accommodated by adding another layer to the preceding scenario like this:



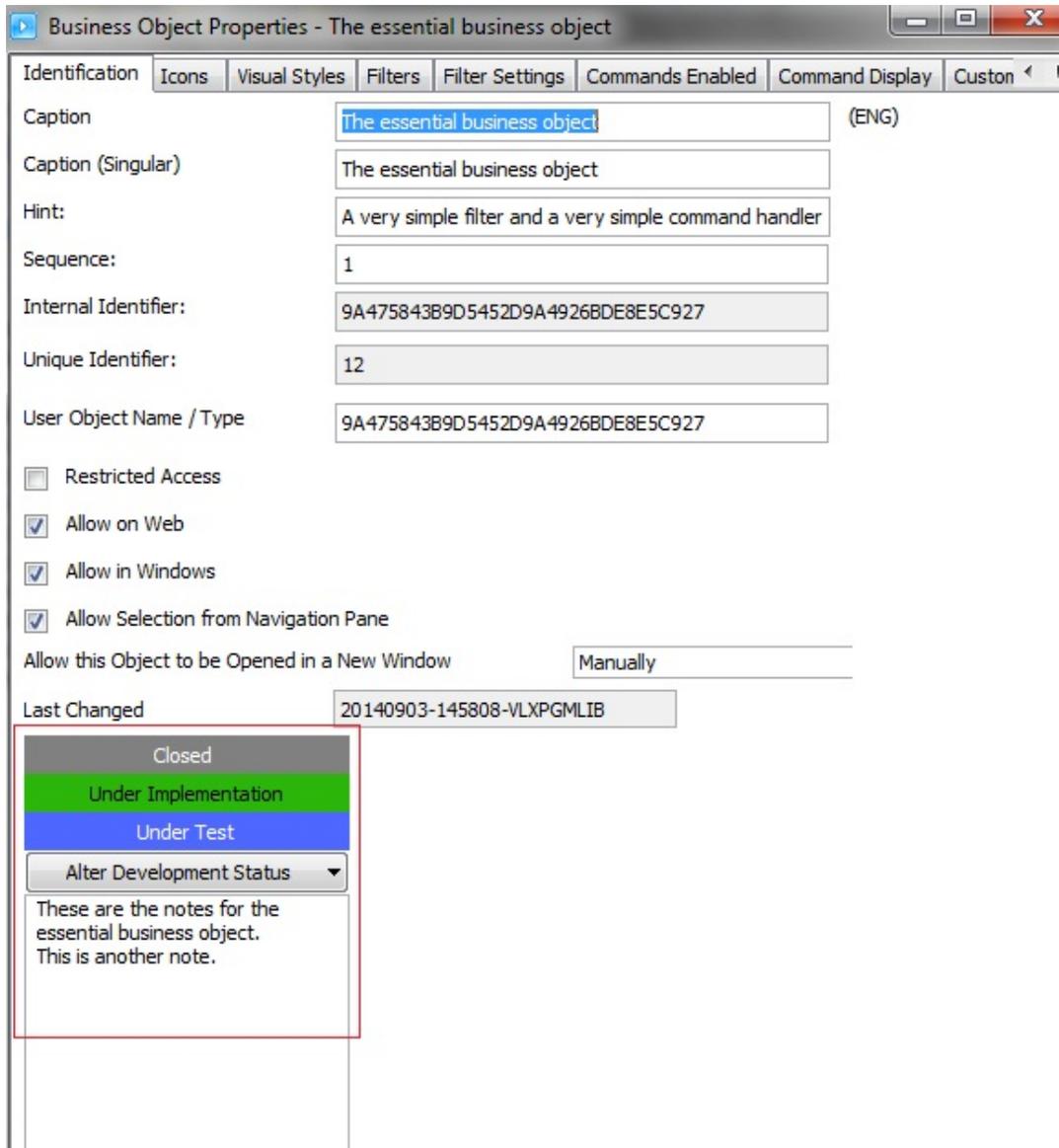
Here three designers and their assigned development teams each individually prototype, code, test and implement parts of the final application using a development cycle like the one defined in [Usage Scenario: One Designer – Multiple Developers](#) .

As application components are completed by each team the designer uses the merge tool to create a merge list of the new and updated components. This is sent to the master designer who merges them into a master Framework. The new master Framework is then sent back to each designer to form the basis on which they should perform their ongoing work.

Development Status Feature

The Development Status feature allows developers to attach development status indicators and notes to various parts of their Framework.

Status indicators and notes can be attached to most Framework objects. They are set on the objects Identification tab:

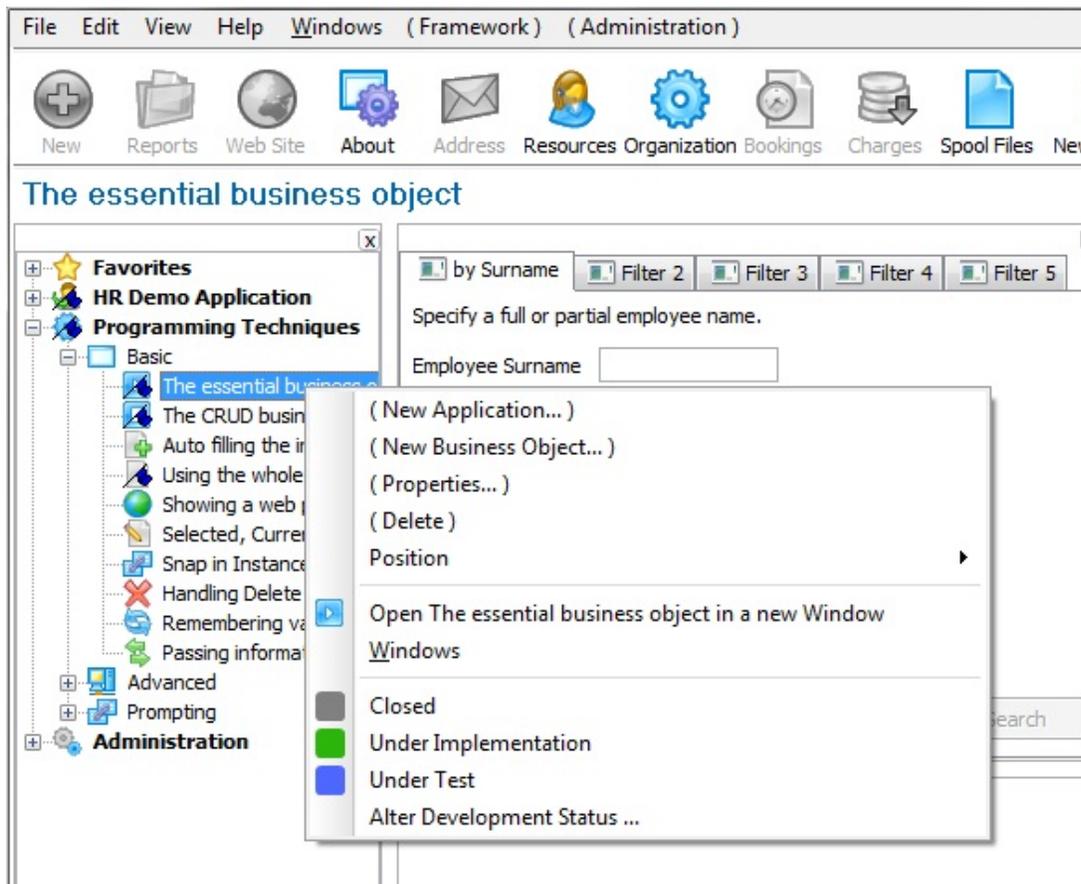


There are ten status indicators to choose from. Simple notes about the development status can also be added.

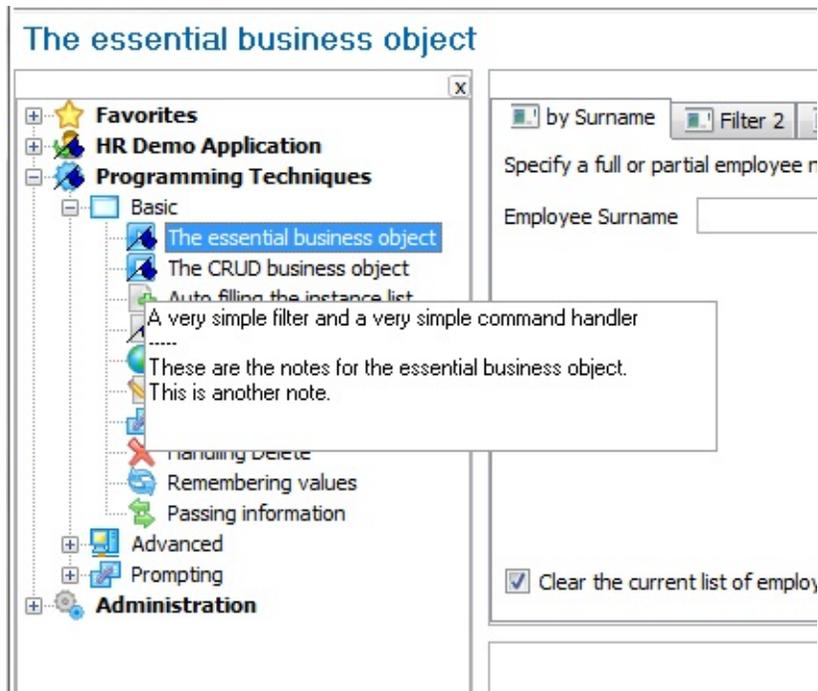
The statuses and notes are stored in the Framework XML schema, so when a Framework definition is exported and shared, other developers can see the status and notes for an object.

When the Framework is executed, the development status and notes are visible if the feature is enabled ([Enable Development Status Feature](#)) and the Framework is run in Render Style M in Development Mode.

When the feature is enabled, the developer can also right-click and use the context menu to see the status of objects on the navigation pane:



They can also see the developer notes (along with the object's hint) when they mouse over a navigation pane item:



When merging frameworks, the merged in development status indicators and notes will overwrite any existing notes and status indicators in the target framework.

How to Create a New Framework Version

Use [Save As](#) to create a new Framework version.

[Associated XML Definition Files](#) (you may sometimes want to change them).

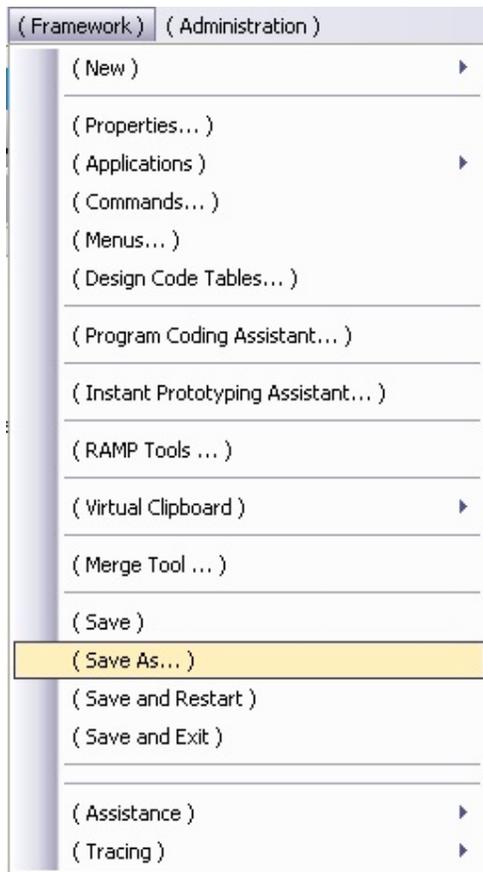
[Start a Framework Version](#)

[Notes For RAMP Users](#)

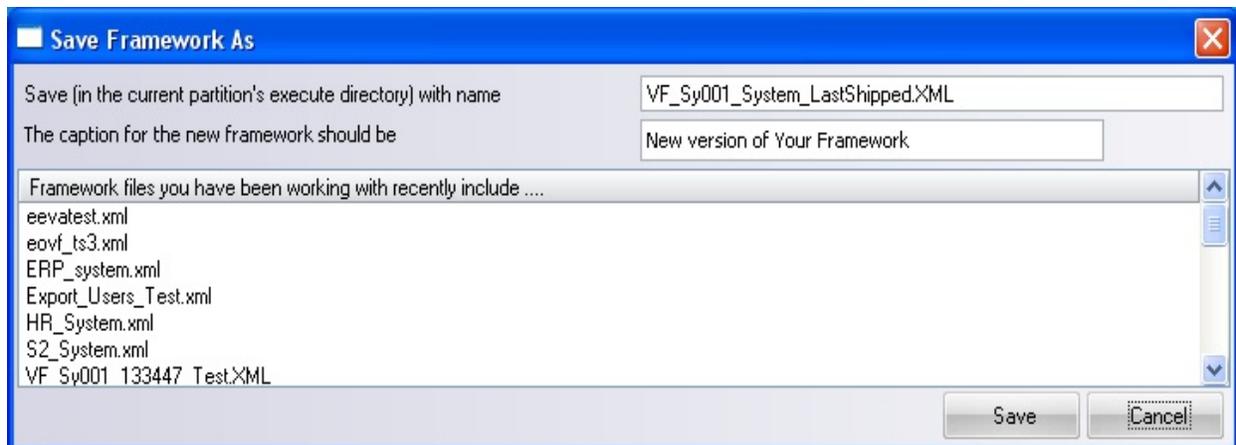
Use Save As

To create a new version of the Framework:

Select the Save As... option of the Framework menu:



The Save Framework As dialog is displayed:



Framework versions are saved as XML files in the Execute directory of your LANSAs program folder.

The default name for a Framework definition is VF_SY001_System.XML.

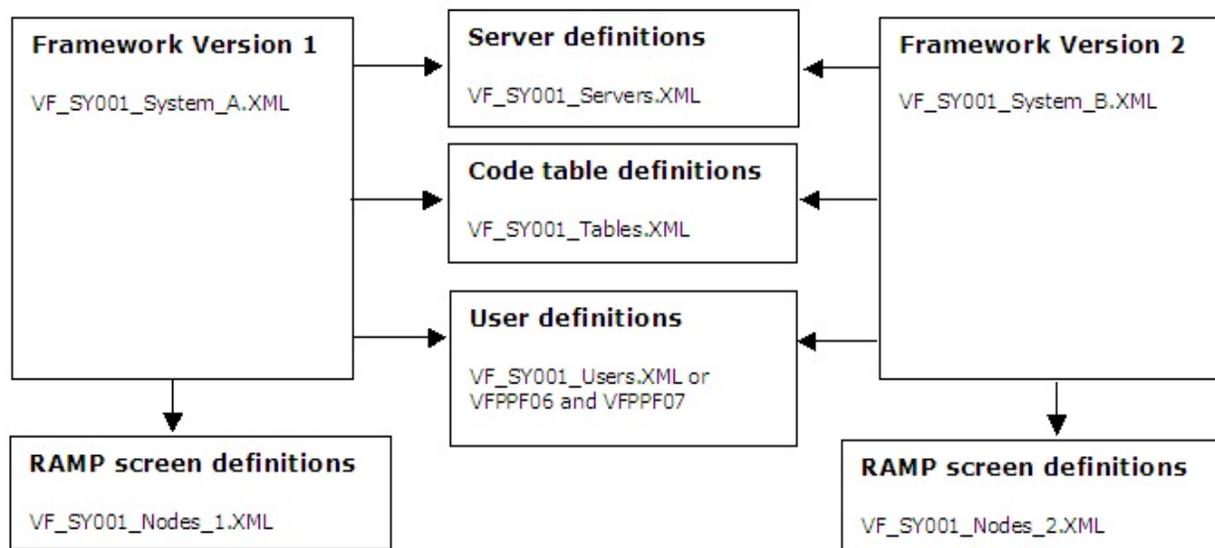
- Change the name of the Framework file.
- Next change the Caption of your newly created Framework version so that later on it will be easy to find out which version of the Framework you are using.
- Click Save to create a new version of the Framework.

Also see [How can I change the list of Framework versions shown?](#)

Associated XML Definition Files

A Framework version has associated XML definition files for servers, code tables, users and RAMP screen definitions (nodes). Several Framework versions can either share these definition files or they can have their own independent definition files.

Typically Framework versions share server, user and code table definitions, but RAMP screen definitions are usually not shared.



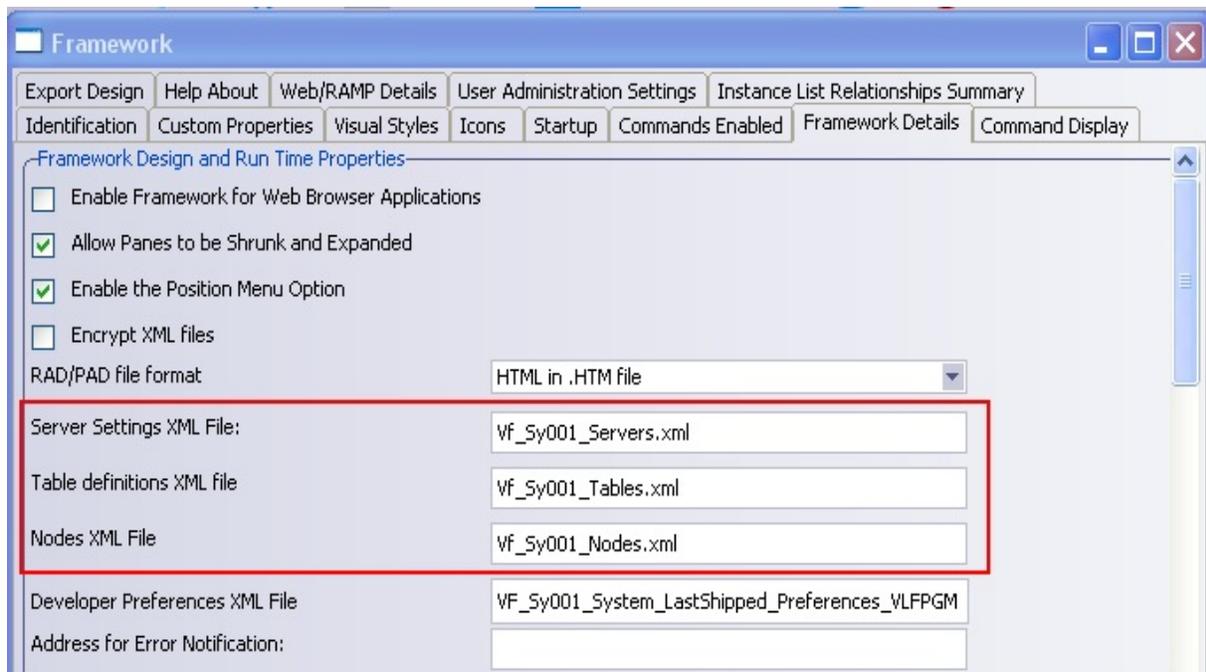
The definition XML files are stored in the Execute directory of the partition you are using.

[Optionally Change Associated XML Definition Files](#)

Optionally Change Associated XML Definition Files

When you have created a new version of the Framework using Save As and you want to use the same associated XML definition files as the original Framework, you do not need to do anything because these details are automatically copied.

If you want to use an independent server, code table or RAMP screen definition, select the Properties... option of the Framework menu and display the Framework Details tab:

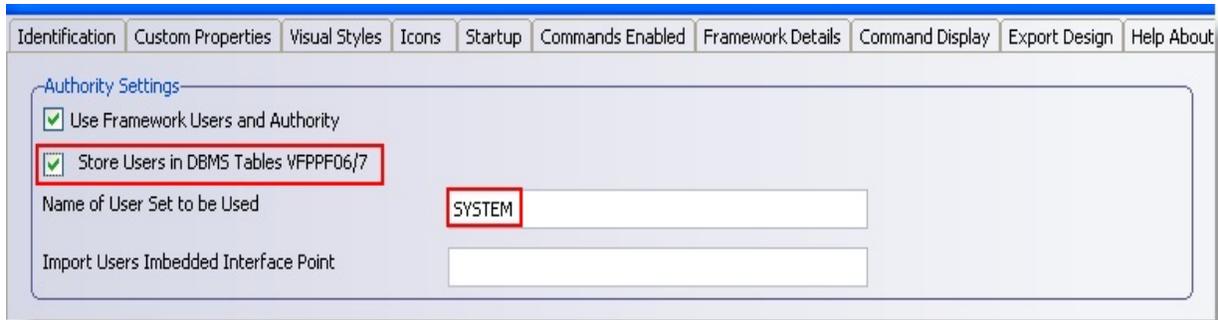


And change the name of the appropriate XML definition files.

Two Ways of Changing User Definitions

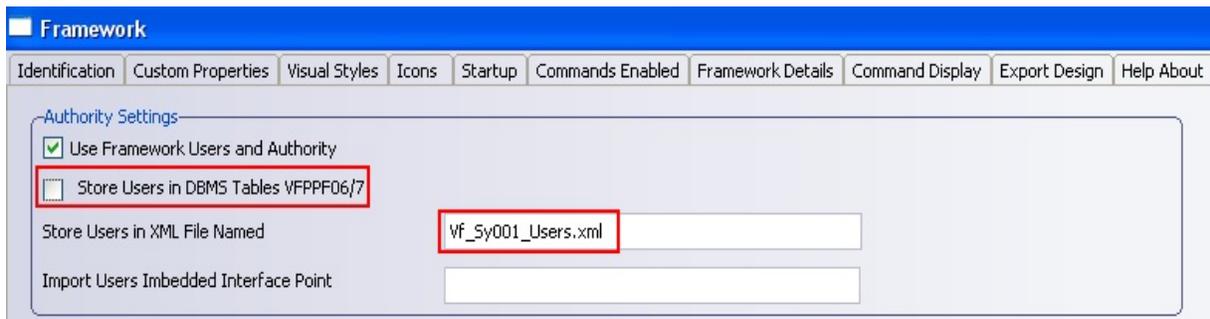
Note that user definitions are a special case. The definitions of the users may be stored in an xml file, but are more commonly stored in physical files VFPPF06 and VFPPF07.

If you want to use different user definitions display the User Administration Settings tab.



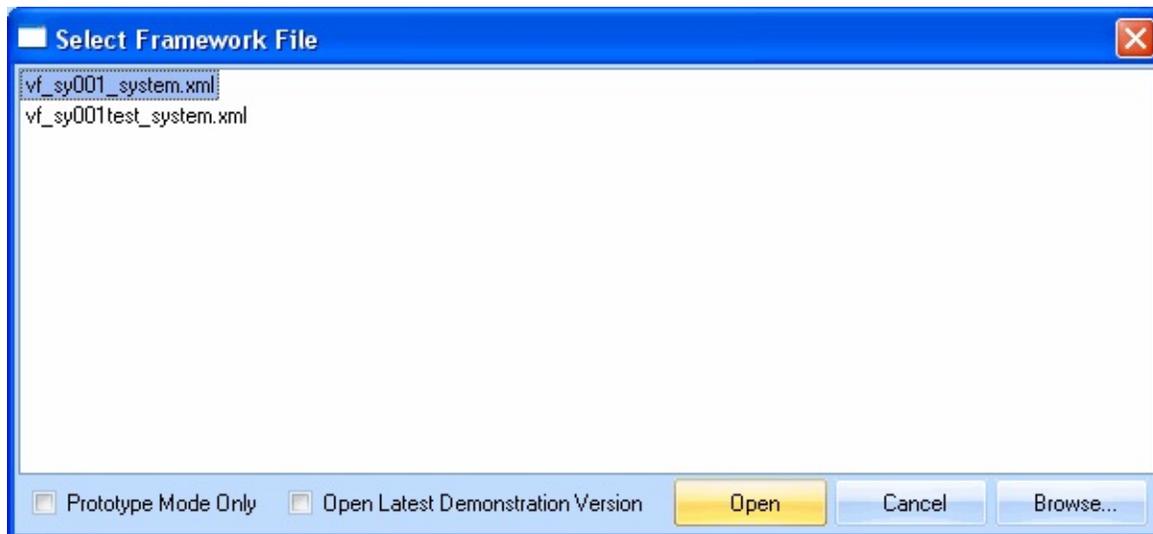
If user definitions are stored in VFPPF06/07 then change the User Set property to a unique value:

If the user definitions are not stored in VFPPF06/07, then change the users XML file to a unique value:



Start a Framework Version

After using the Save As option, subsequent start ups of the Framework will show a selection dialog of the available Framework versions:



The list of XML files comes from the previous Save As copies you have created. The last XML file you used is automatically selected.

Question: Where does this list of XML files come from?

Answer: The list of Save As files is kept in text file in your partition execute directory.

Typically the name of this file is "vf_sy001_system_choice.txt". You can edit it using Notepad.

Question: Where is the name of the last XML file used kept?

Answer: The name of the XML file that you last opened is kept in a text file in your temporary directory.

Typically the name of this file is "vf_sy001_system_choice_Last_Used.txt". You can edit it using Notepad

Notes For RAMP Users

Note that if there are several versions of a RAMP application, at the end of the design-development cycle also the RAMP screen definition files (nodes) need to be merged together manually.

How to Keep Automatic Backup Copies of Your Framework

If you want a back-up copy of your Framework design to be created every time you save the Framework, select the option the [Keep XML File Versions](#) in the Framework Details tab.

The back-up copies of your definition files are kept in the Execute directory of your partition. They have names like vf_Sy001_System_YYYYMMDD_HHMMSS.xml. The last portion of the name reflects the date and time they were saved.

To use a back-up copy, rename or delete the current definition XML file using Windows Explorer, and then remove the date and time portion from the name of the back-up file.

Back-up copies are kept for your [Associated XML Definition Files](#) as well as the Framework definition files.

If you want to be periodically prompted to save the Framework, specify a time interval in the [Automatic Save Time](#) option.

How to Export Framework Definitions

You can exchange Framework designs with other people.

You might send them to other developers to show them how you want filters or command handlers coded. You might also send them to end-users so that they can execute them as way of validating your design proposal.

The sender should:

- Use the export facility to create a zip file in the Export Design tab of Framework properties.
- Send the zip file to the receiver(s).

The receiver should :

- Export their existing Framework design before installing the received Framework design.
- Unzip the received Framework into the partition's execute directory.
- Work with the received design.

To restore the original Framework design, unzip the saved Framework.

Note that distributing the Framework is not the same as to [Deploy the Application](#).

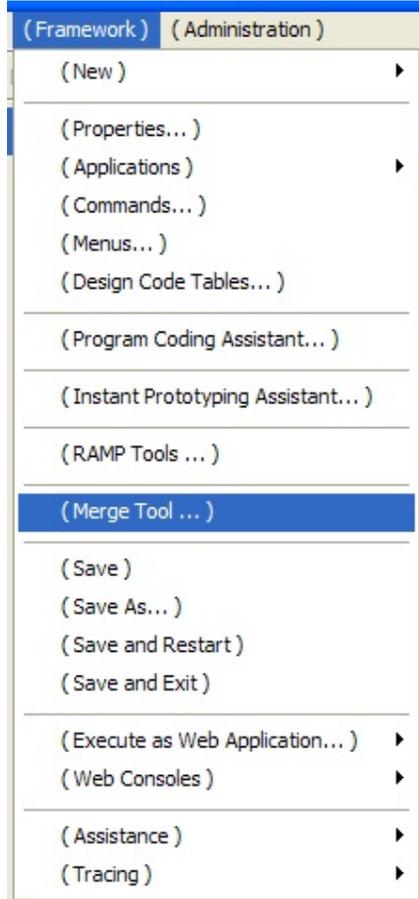
How to Merge Items from one Framework to Another

The VLF merge tool works with merge lists which are lists of Framework objects. You can:

- Create a merge list of items from your Framework and send it to someone.
- Receive a merge list from someone and merge the items into your Framework.
- Transfer items between different Frameworks which reside on your PC.

The merge tool has been primarily designed to assist multiple designers and multiple developers to concurrently develop a single master Framework. They do this by continually merging new or modified Framework objects back into a single master Framework.

You can start the merge tool when using a Framework as a designer like this:



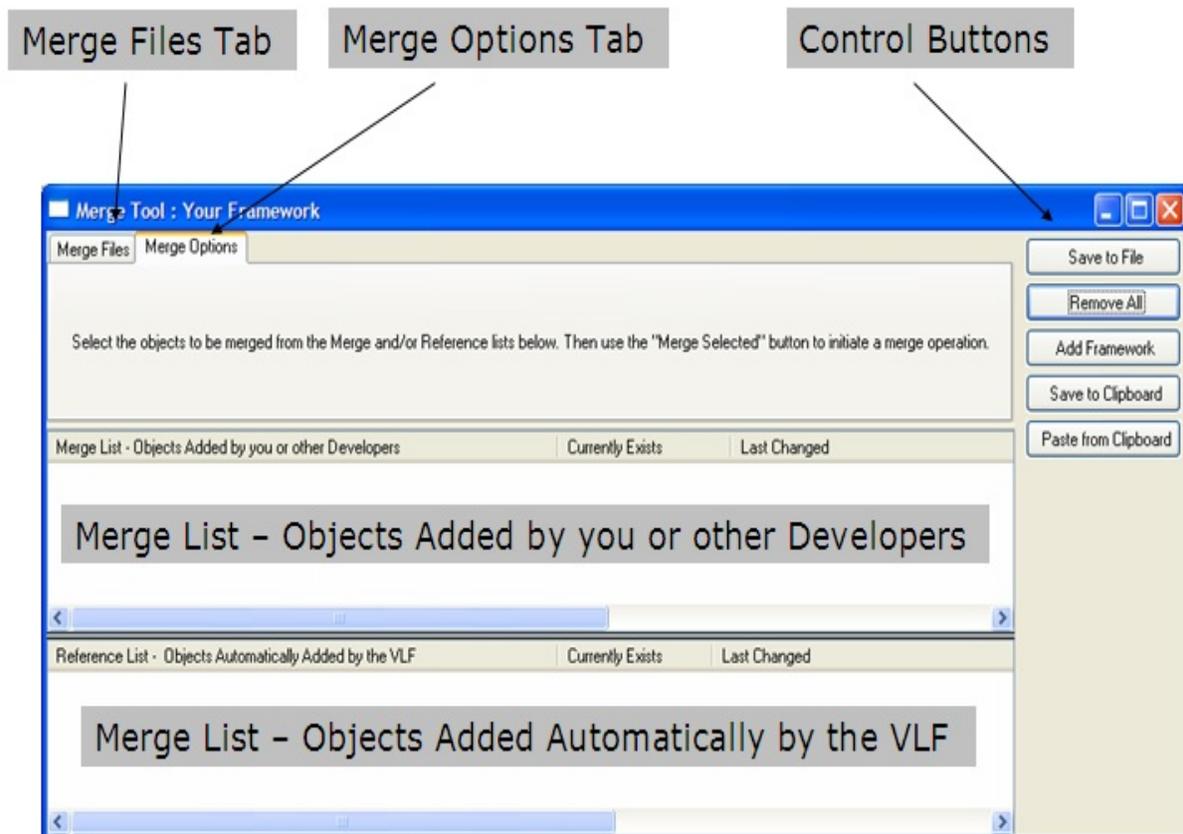
[The Merge Tool Window](#)

[Using the Merge Tool](#)

[Example of Combining Two RAMP Framework Definitions](#)

The Merge Tool Window

The Merge Tool window looks like this:



Merge Files Tab

This tab displays a temporary folder on your hard drive containing merge files. You can:

- Drag and drop merge lists into this folder (for example from a received e-mail).
- Drag and drop merge lists out of this folder (for example into an e-mail you are about to send or onto a shared network drive).

Merge Options Tab

This tab is used when you have received a merge list and are going to merge some of the things that it contains into your current Framework.

Merge List – Objects Added by you or Other Developers

This displays the Framework objects that you (or some other developer) have manually added to the merge list.

Merge List – Objects Automatically by the VLF

This displays the Framework objects that the VLF has automatically included into the merge list because they are referenced in some way by the objects that your or some other developer added manually. Essentially this part of the merge list exists to help you avoid sending incomplete merge lists to someone else.

Buttons

Save to File	Saves the current merge list into a file so that it can be sent to someone else.
Remove Selected	Removes the selected items from the merge list. Used to remove items inadvertently added to a merge list.
Remove All	Used to clear the entire merge list.
Add Framework	Adds the definition of the current Framework to the merge list. This option is a button because you cannot drag and drop the Framework itself onto the merge list area.
Save to Clipboard / Paste from Clipboard	Saves or restores the current merge list using a local clipboard-like file. Use this option when you are merging items between different Frameworks that reside on the same PC. For example, to transfer Framework items between Frameworks A and B resident on the same PC you would do this: Open Framework A, create a merge list of the required items, then use Save to Clipboard. Then open Framework B and click Paste from Clipboard. Select the required items and merge them into Framework B.
Merge Selected	Starts a merge operation using all the selected items from the merge list. When clicked the Merge Options tab will be displayed and request further options and choices be made to initiate the merge operation.

Using the Merge Tool

How to Send a Merge List to Someone Else

- Start the merge tool
- Drag and drop the objects from your Framework that you want to send into the merge list. For example, you might drag and drop the 3 new business object filters that you have added to your Framework.
- Use the Save to File button.
- From the Merge Files Tab select the file that was saved and drag and drop it into the place that will be used by the recipient to receive it. For example, to a shared network drive, or into an e-mail message, etc.

How to Receive a Merge List from Someone Else

- Start the merge tool
- Switch to the Merge Files Tab.
- Drag and drop the merge list file you received into the Merge folder (for example from an e-mail message, from a shared network drive, etc).
- After a few seconds the Merge Tool will prompt you to open the file you have just dropped. Click Yes. The contents of the merge file will then be displayed in the Merge List – Selected Items and Merge List – Associated Reference Items areas.
- Select the objects to be merged into the current Framework from the Merge List areas by clicking on them. Then click on the Merge Selected button. Use the Merge Options tab to control how they are merged into your Framework.
- Remember that you can and should be selective about what you merge. You do not have to merge in everything that someone has sent you.



Some Useful Details

- All Framework items (or objects) are uniquely identified by a GUID (Global Unique Identifier). Theoretically a GUID is a 32 character identifier that is unique in space and time. No two GUIDs are ever the same. This means if that developer A creates a business object captioned "Customer" and developer B creates a business object captioned "Customer" they are not the same Framework object. They are two completely separate business objects that just

happen to have the same caption. In the Merge List display area is a column titled "Currently Exists". This indicates whether an object of this type with this GUID currently exists in your Framework.

- The benefit of the GUID approach is that objects from any two Frameworks in the world can be merged without risking object or property duplication.
- The downside is that two developers at the same site cannot create a single object if they both create an object of the same type with the same caption because the GUIDs of these two objects will be different.
- The preceding points mean that the optimal project work flow model is probably one where the main designer always creates new high level "container" objects such as applications, business objects, 5250 RAMP sessions, etc. These are then sent out to the individual project developers, who then extend and enhance them as required, sending back their alterations periodically. This way all the high level container objects being used all have the same GUID (ie: they are actually the same object) which makes merging them simpler and easier to manage.

Example of Combining Two RAMP Framework Definitions

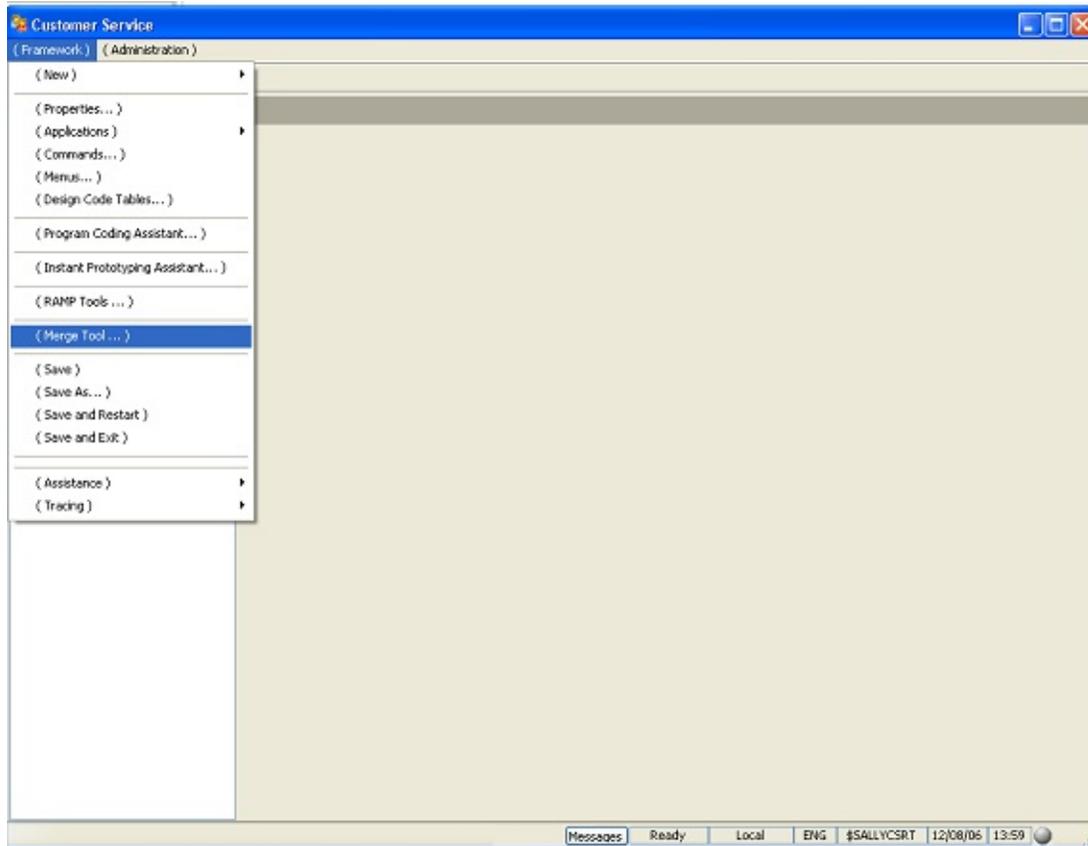
In this example we will merge developer B's Framework and RAMP definitions into Developer A's Framework definition.

[Collecting the Definitions](#)

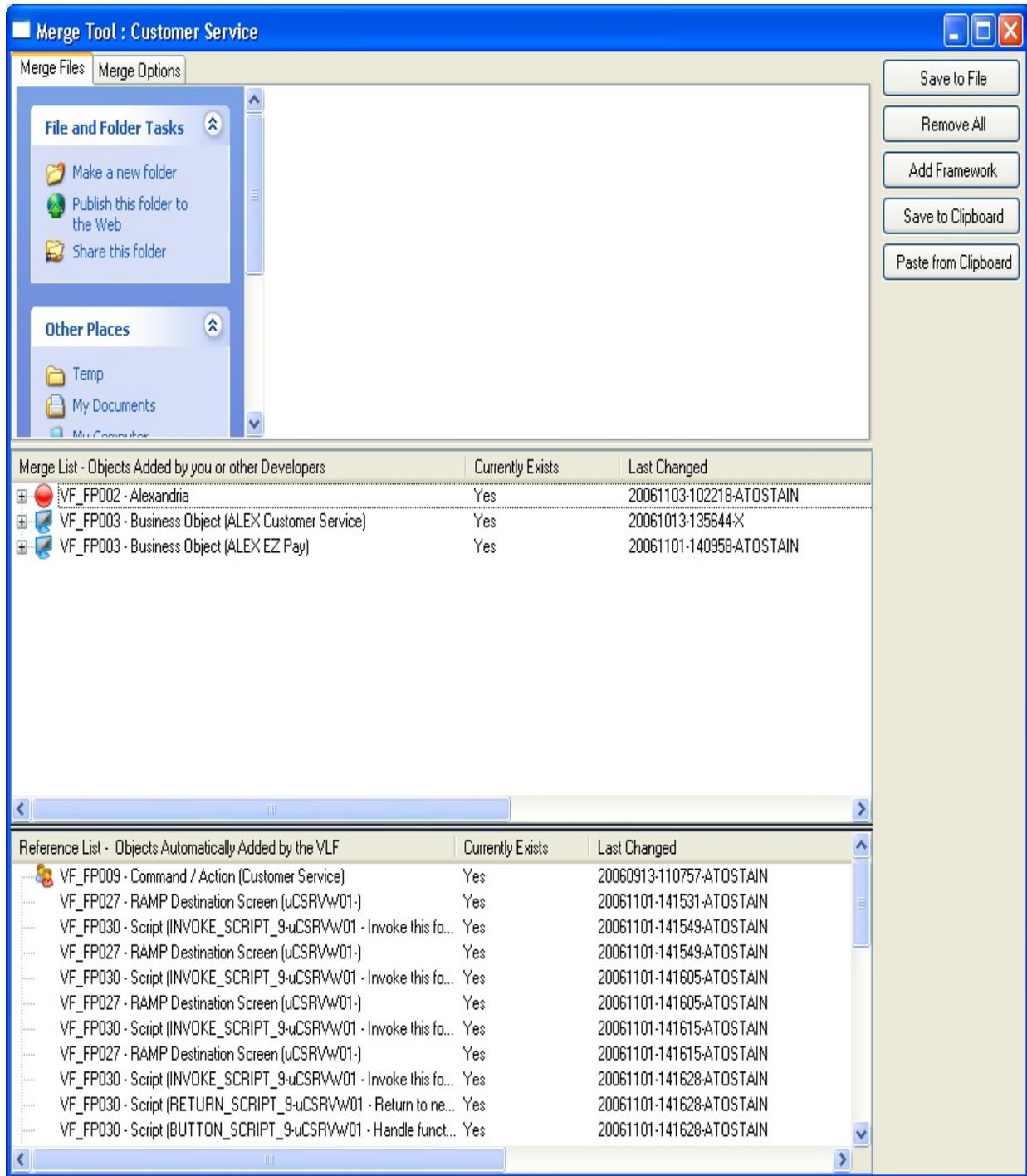
[Merging the Definitions](#)

Collecting the Definitions

On Developer B's PC, start the Merge Tool:

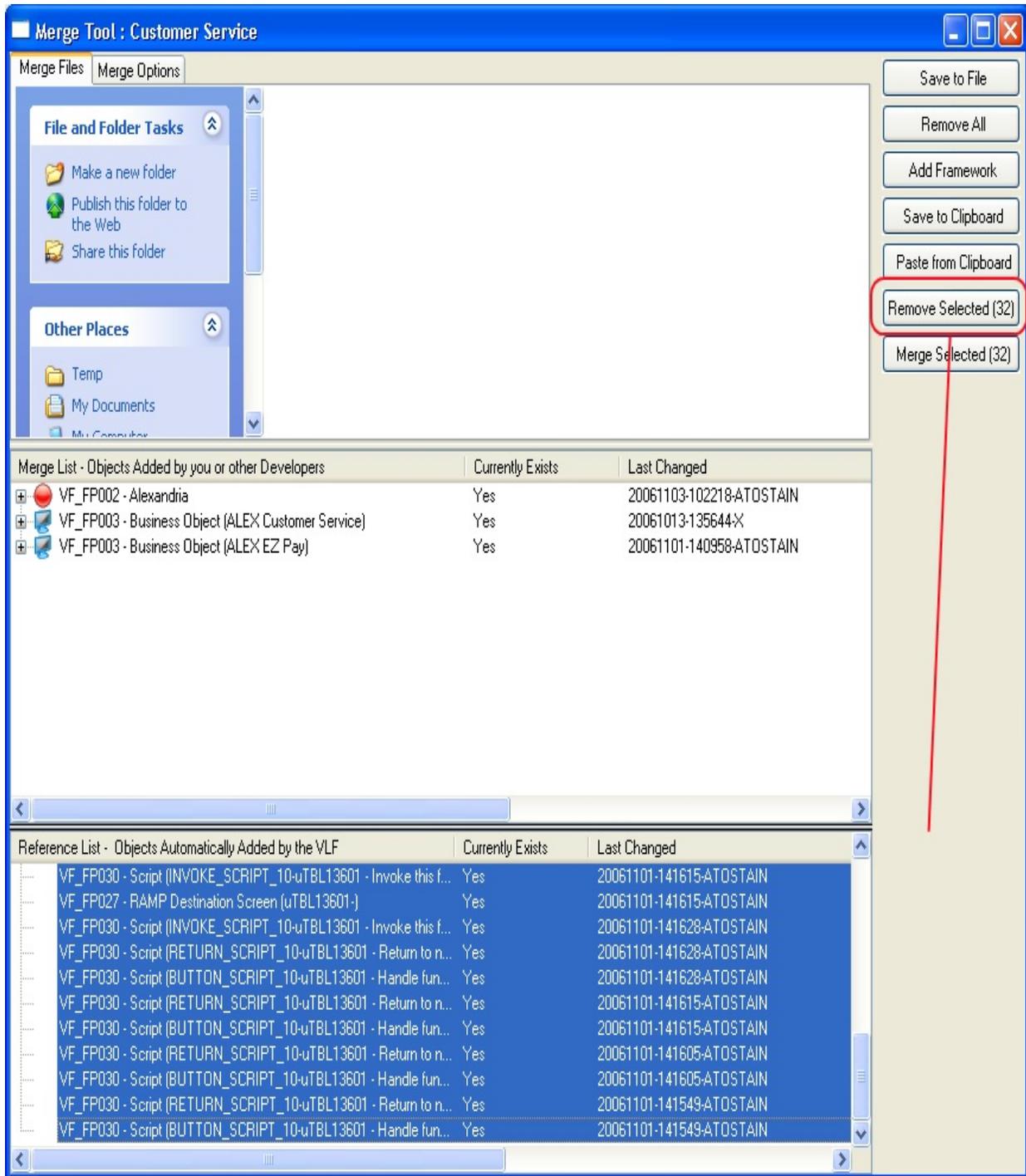


Click and drag the business objects from the Framework window to the middle panel, the Merge List. In this example, we are dragging Alexandria:

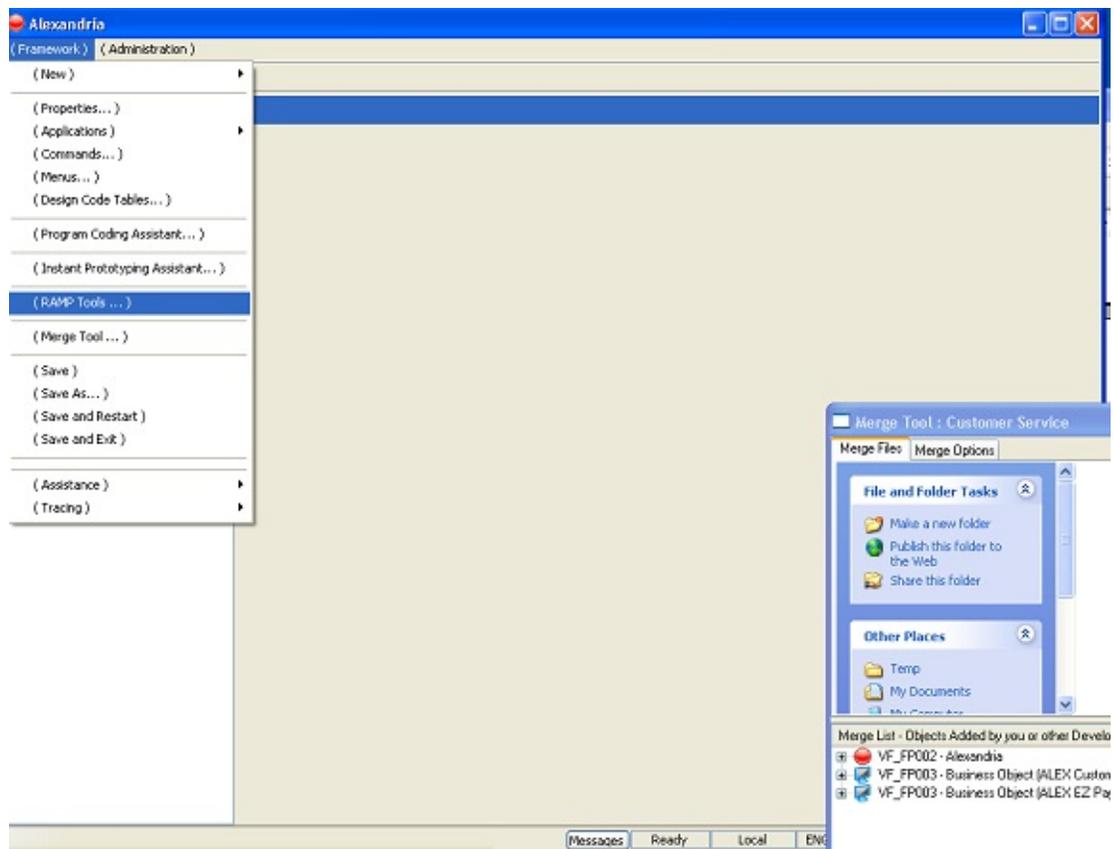


Note that the bottom panel contains all of the objects that the merge tool considers required as well.

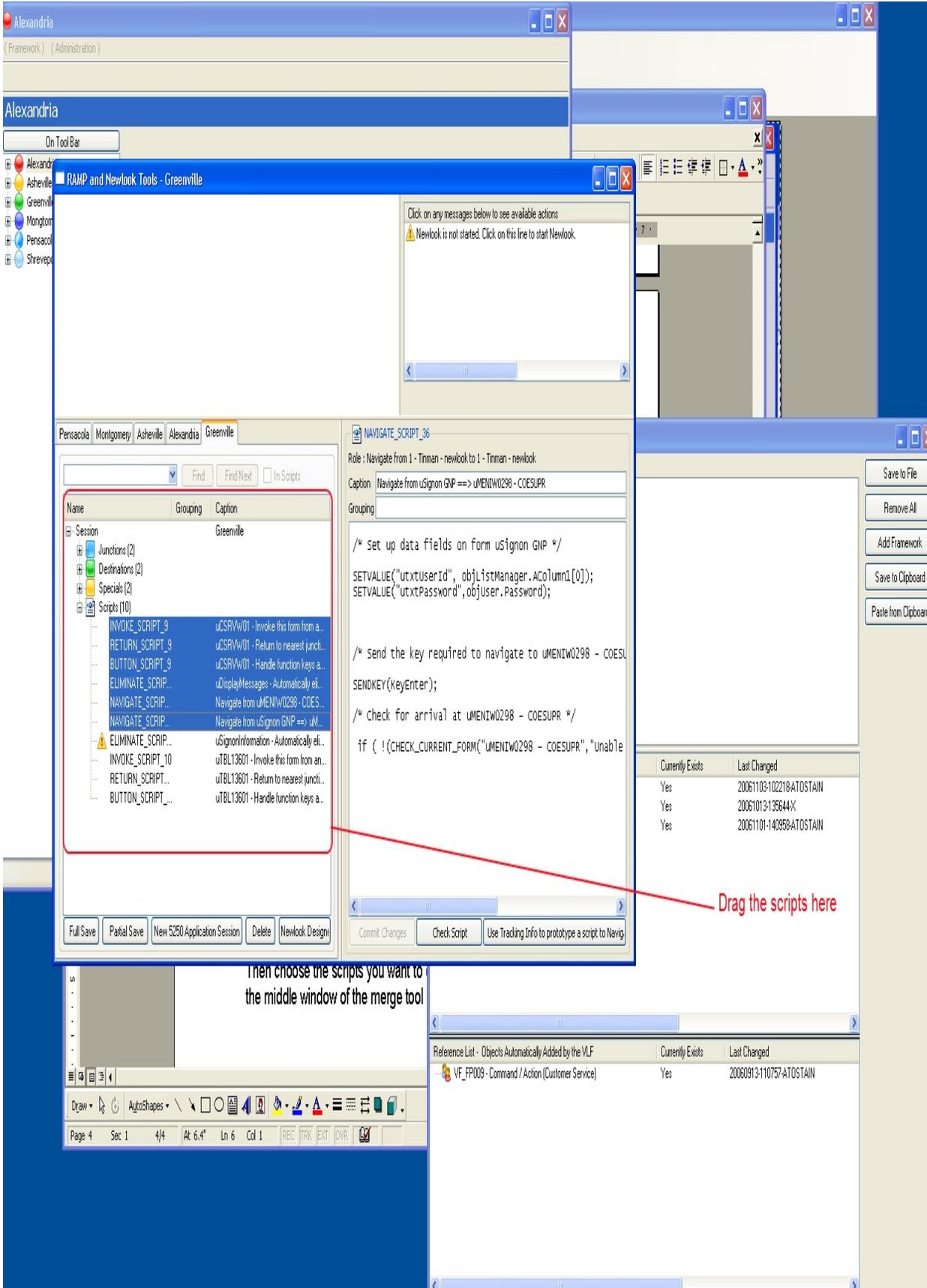
The merge tool sometimes will bring too many objects over. You may wish to delete the scripts not required here and manually copy the ones you will need manually. Highlight the scripts and click Remove Selected.



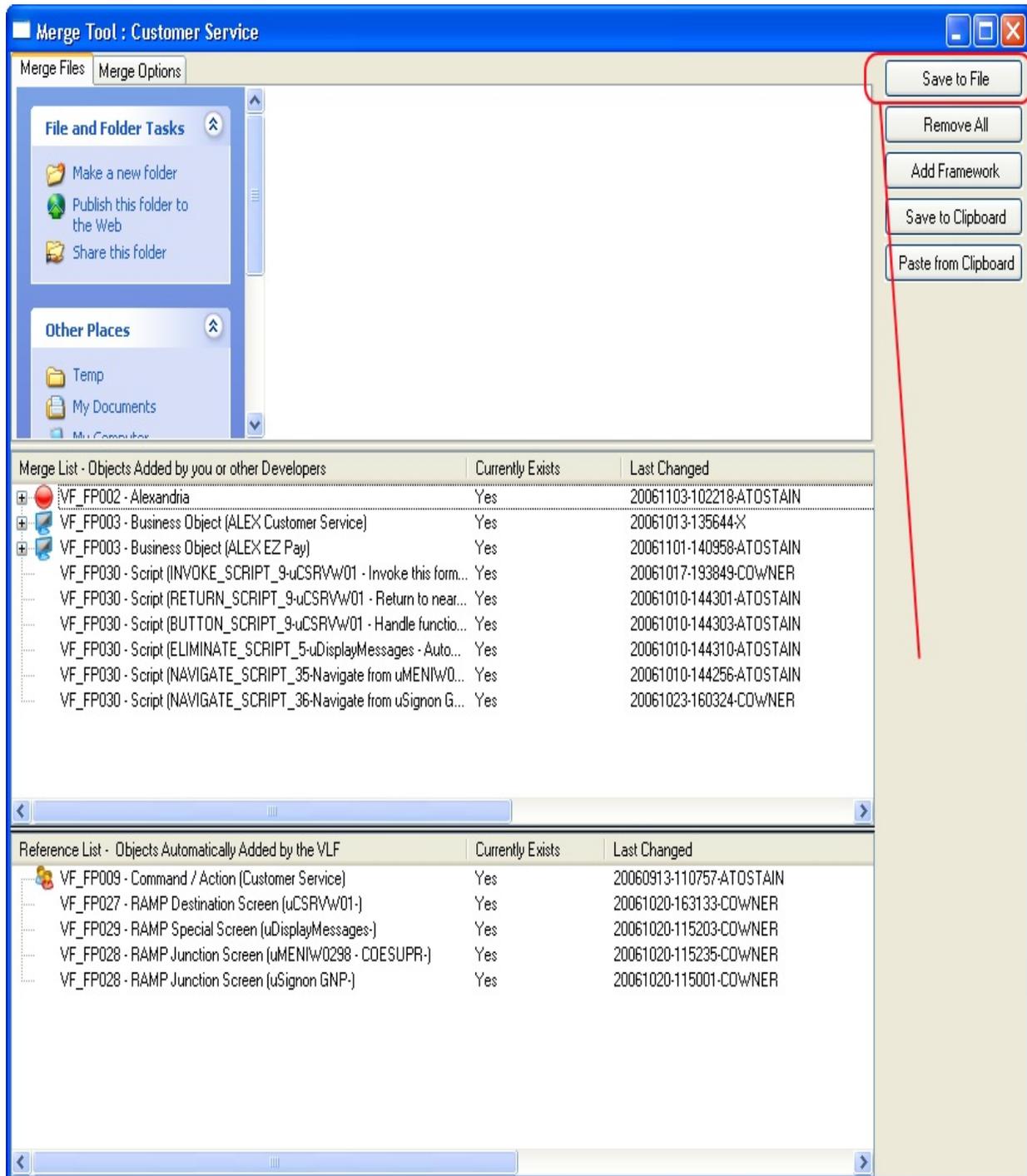
Then start RAMP Tools.



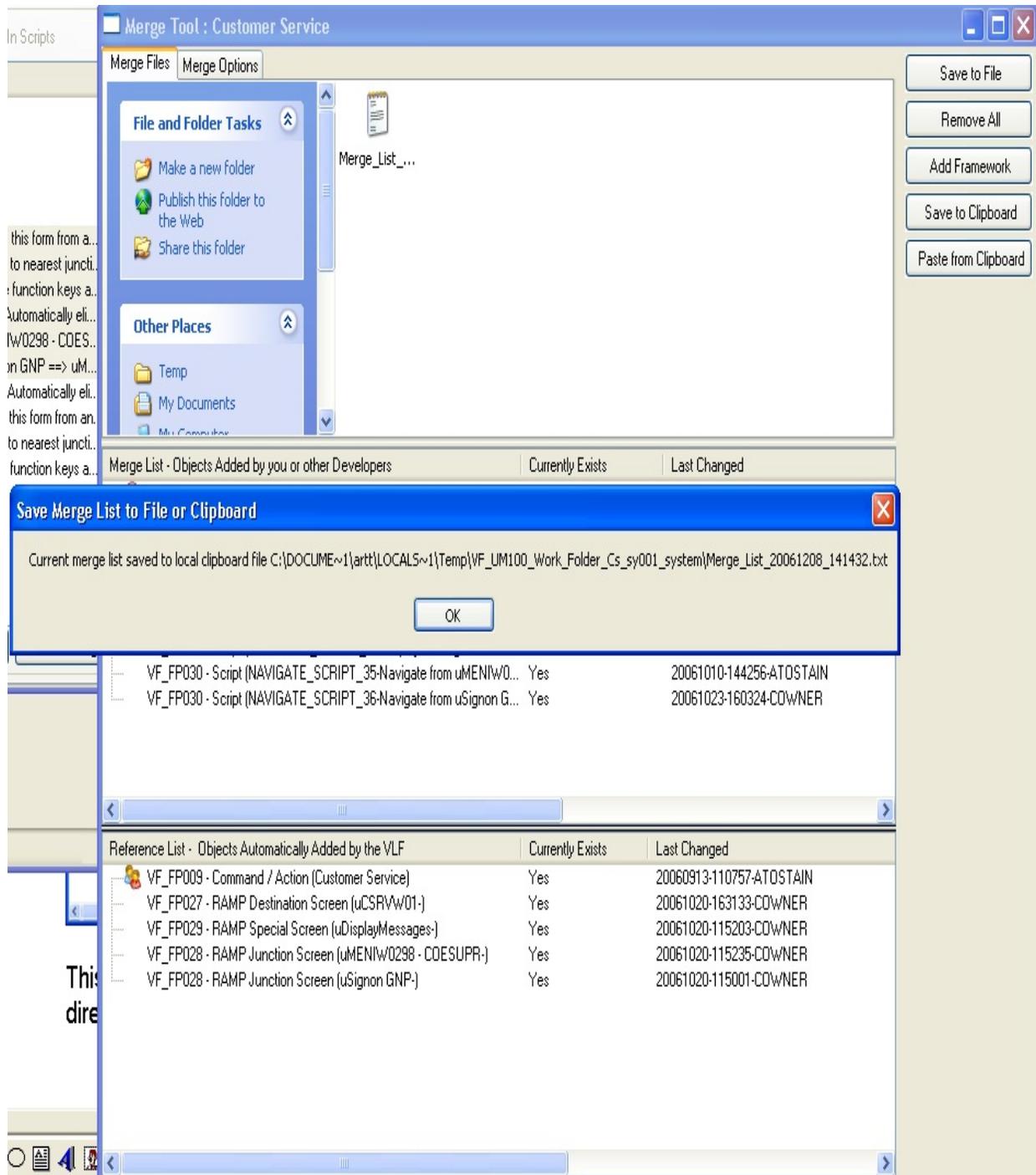
Then choose the scripts you want to copy, and click and drag them to the middle window of the Merge Tool.



When all of the objects are pasted, click Save to File button.



This will create a file containing the objects into the temporary directory.

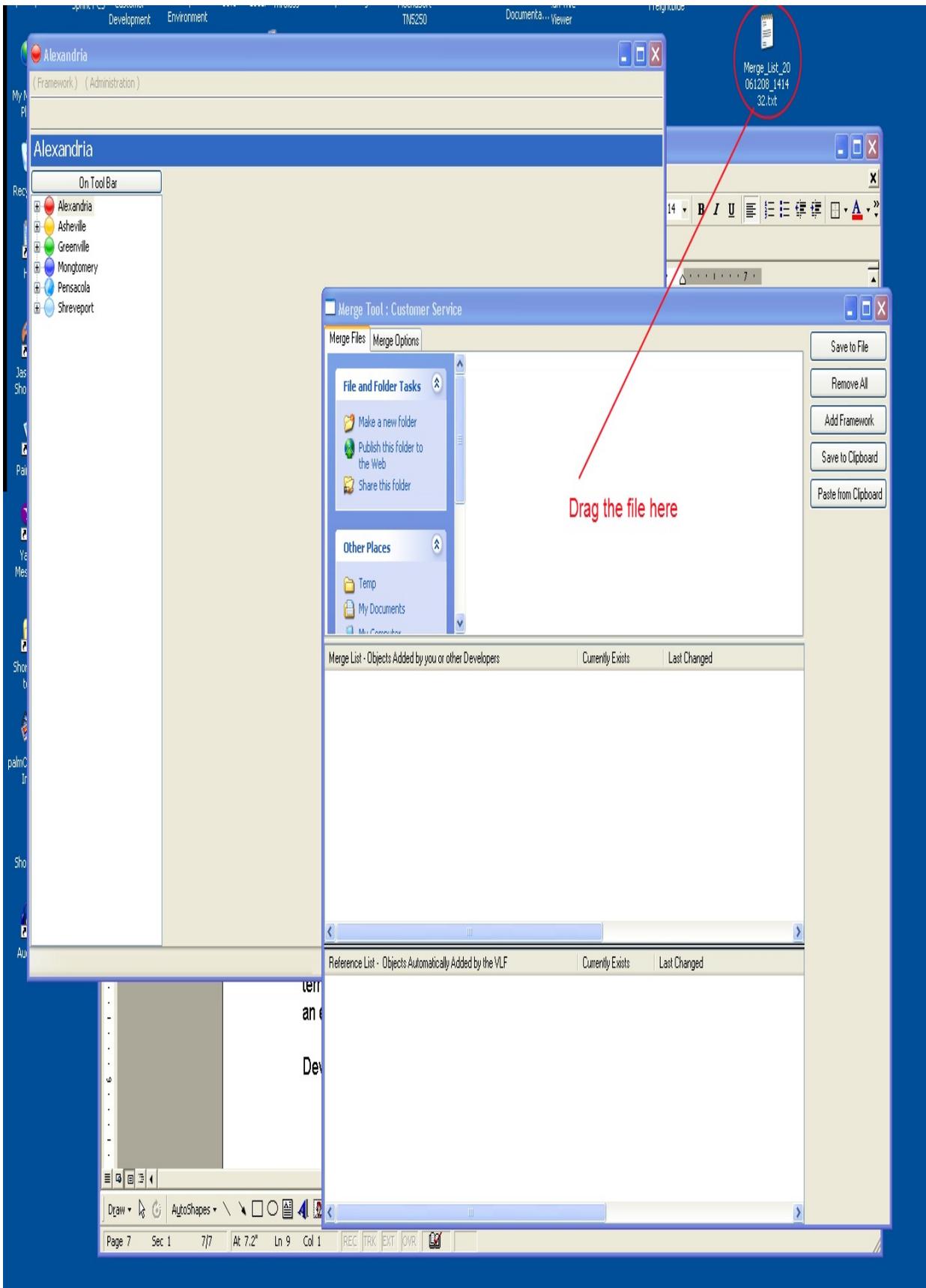


Send this file to Developer A.

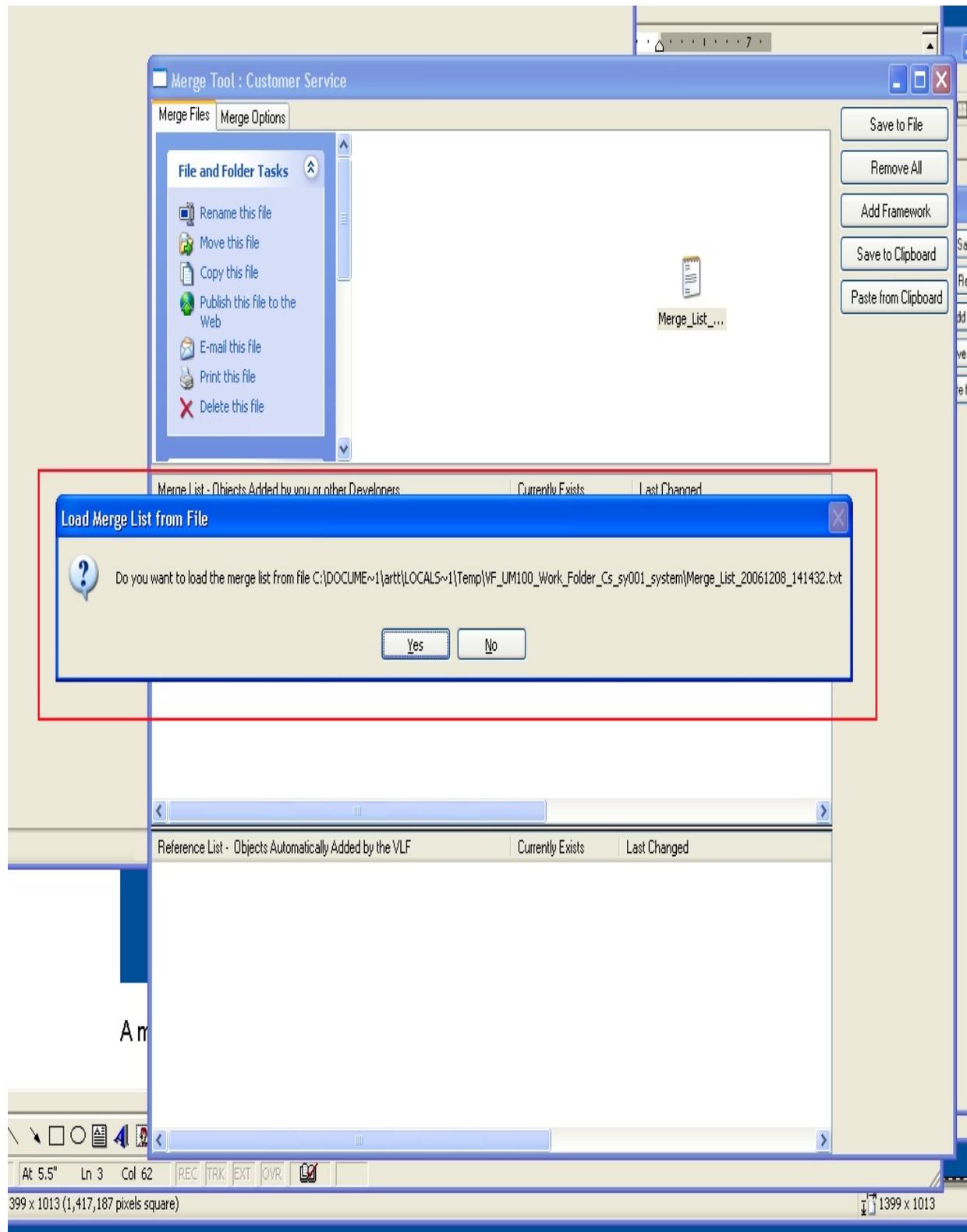
Merging the Definitions

The file is saved in the developer's temporary directory. You can also use Windows to drag the file onto an email.

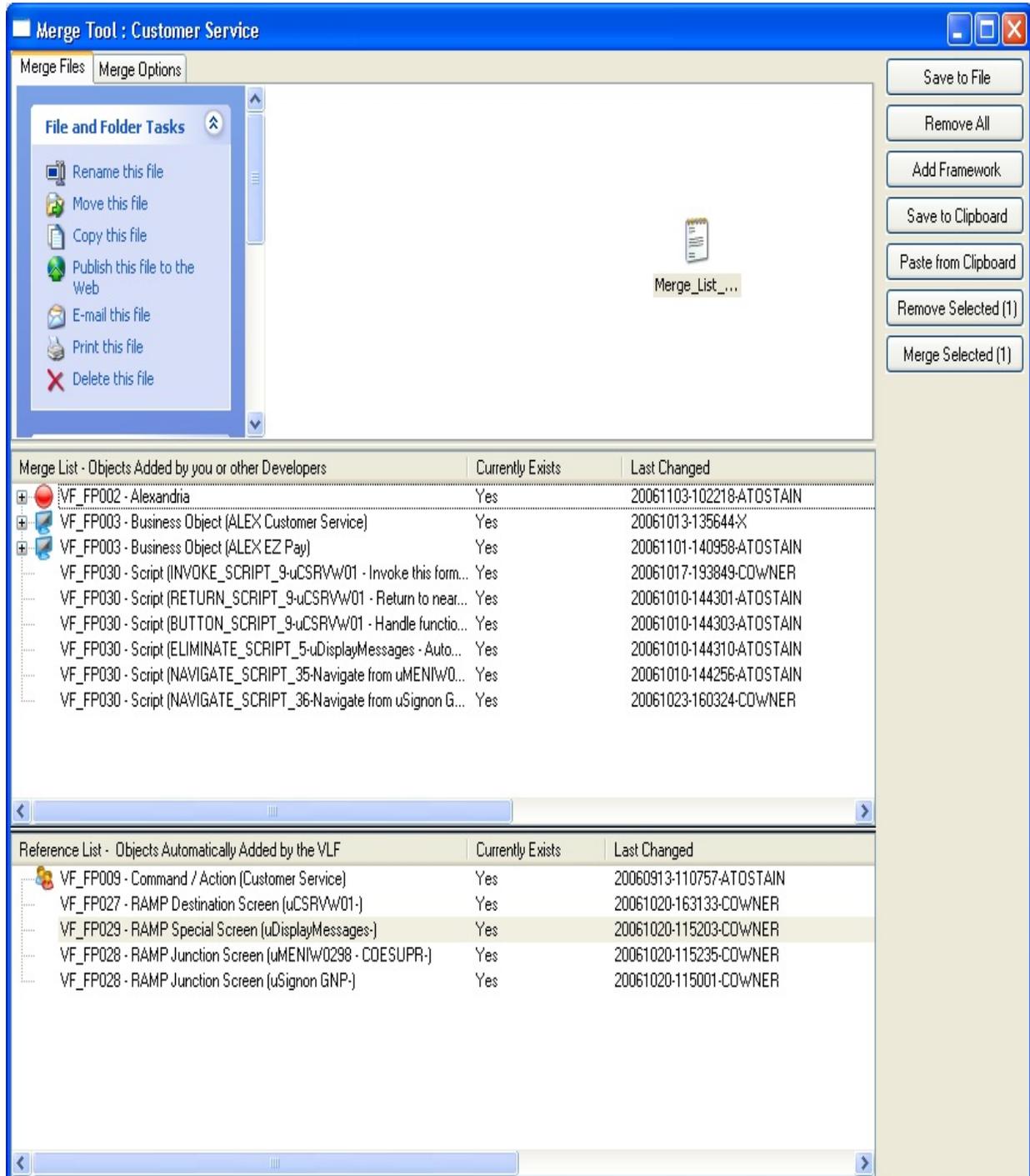
Developer A will drag this file into the top panel of the Merge Tool.



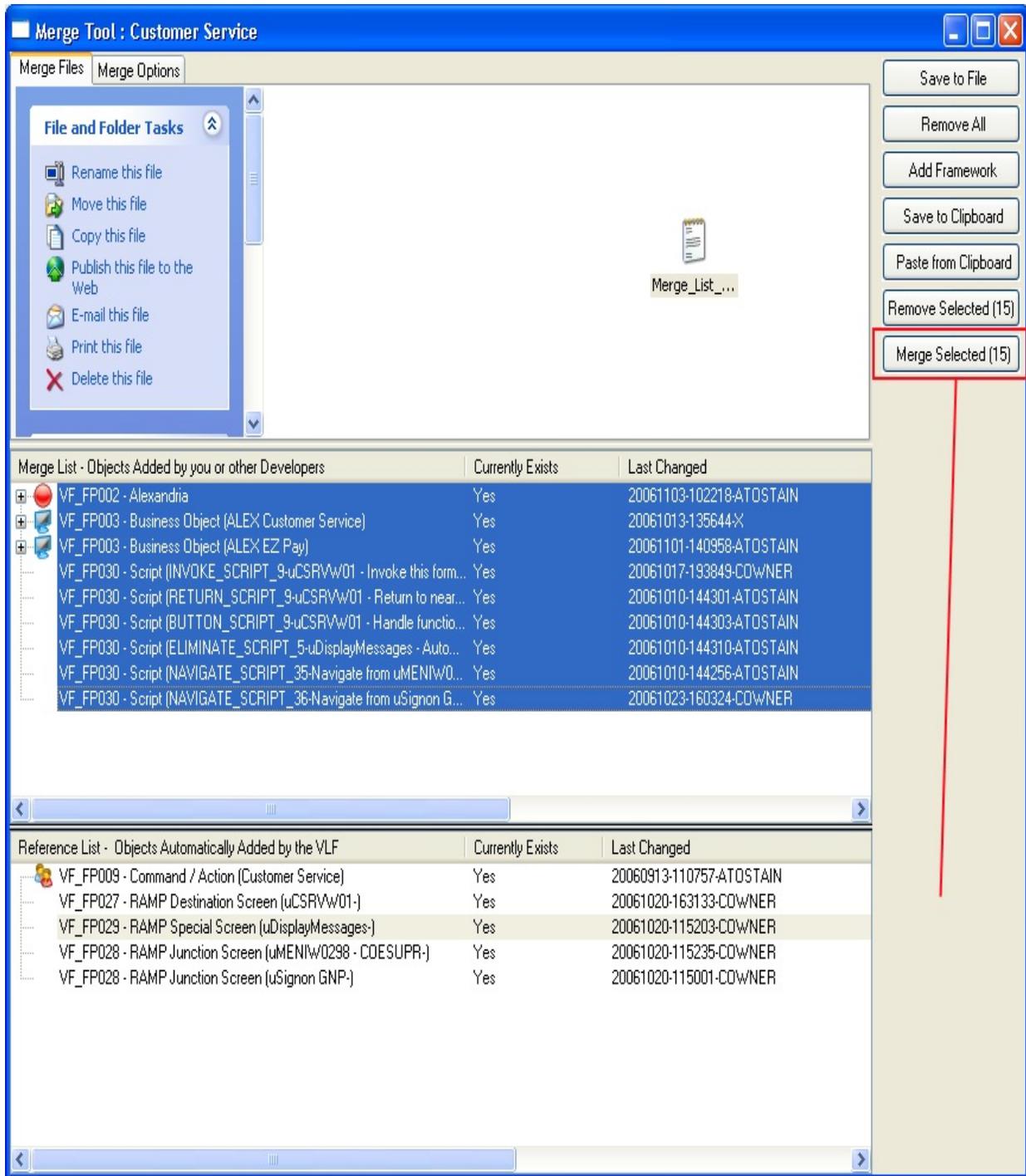
A message will pop up asking you to confirm the file selection.



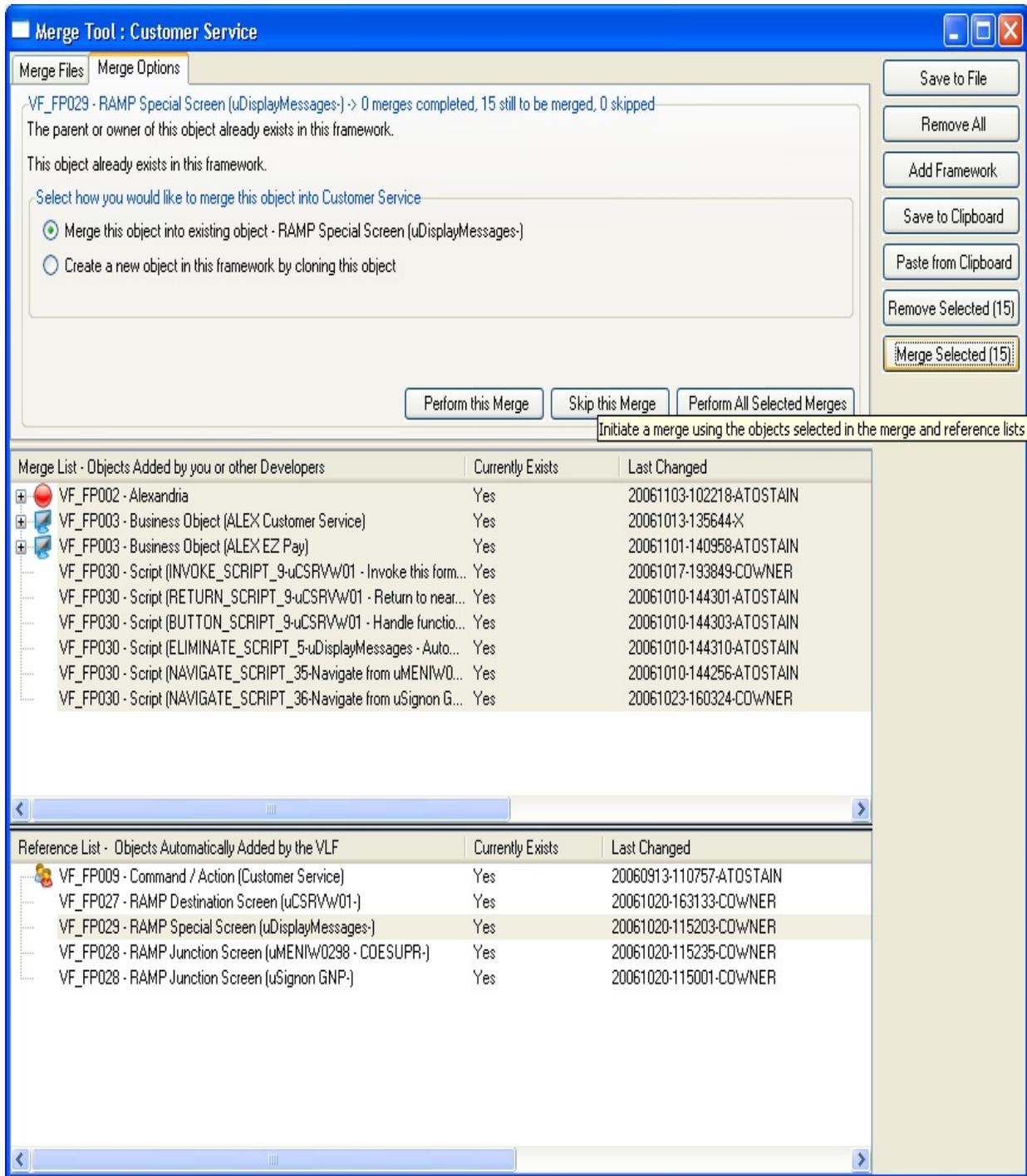
Click YES to confirm.



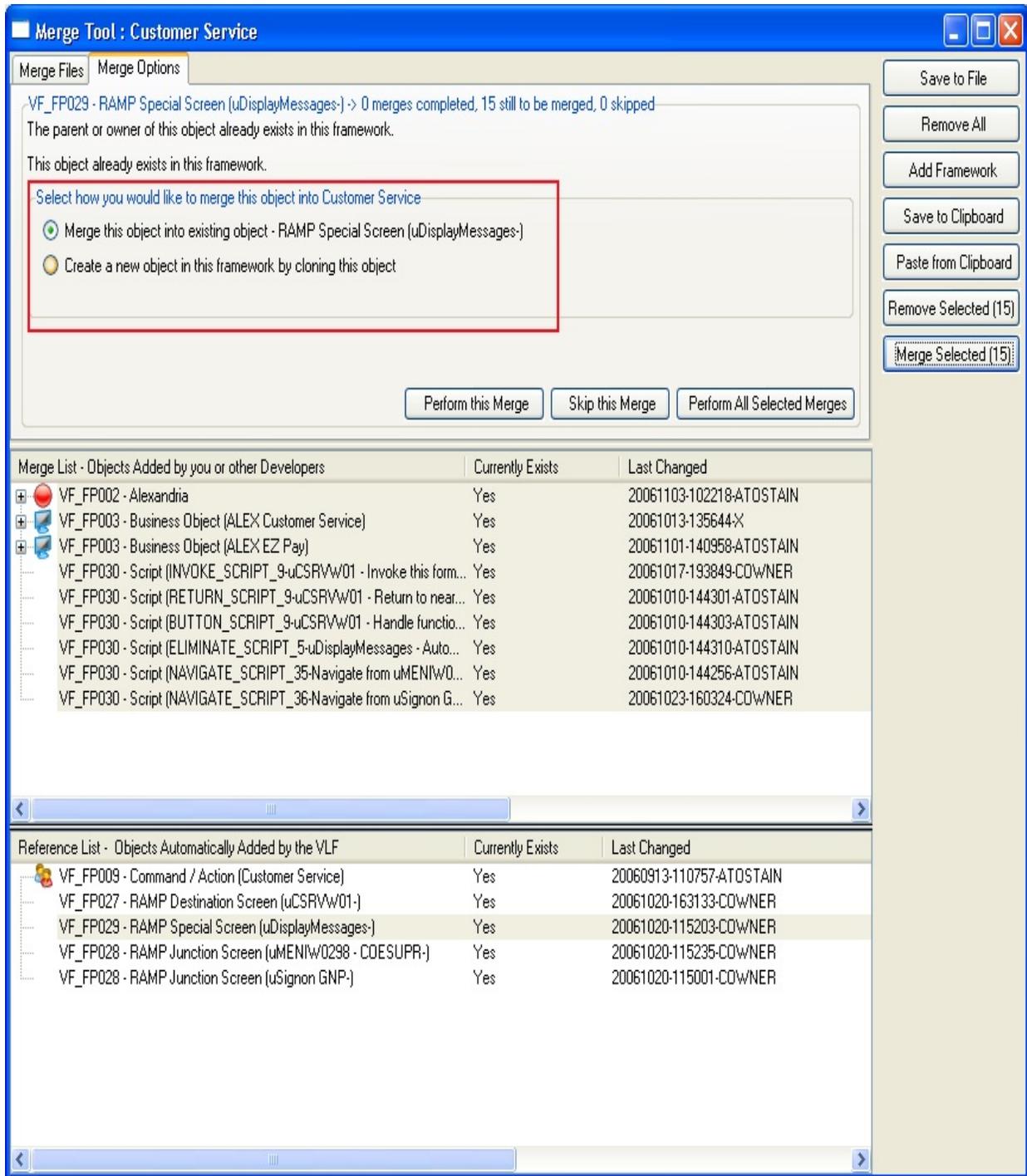
Highlight the components you want to merge in the middle panel, then click Merge Selected.



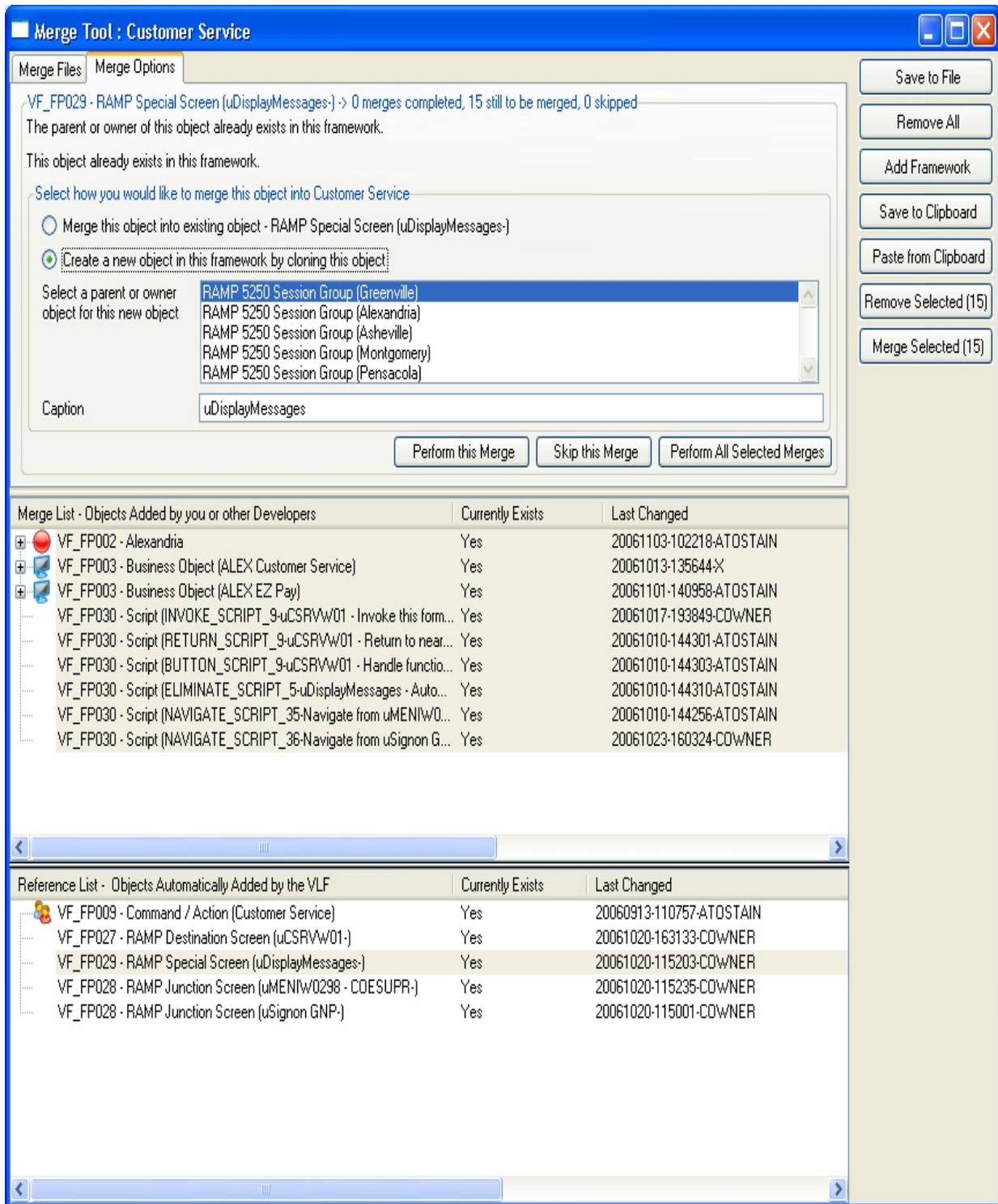
The screen will change:



Specify whether to merge into an existing object or create a new one.



If you choose Create a new object, you will be asked to name the object:



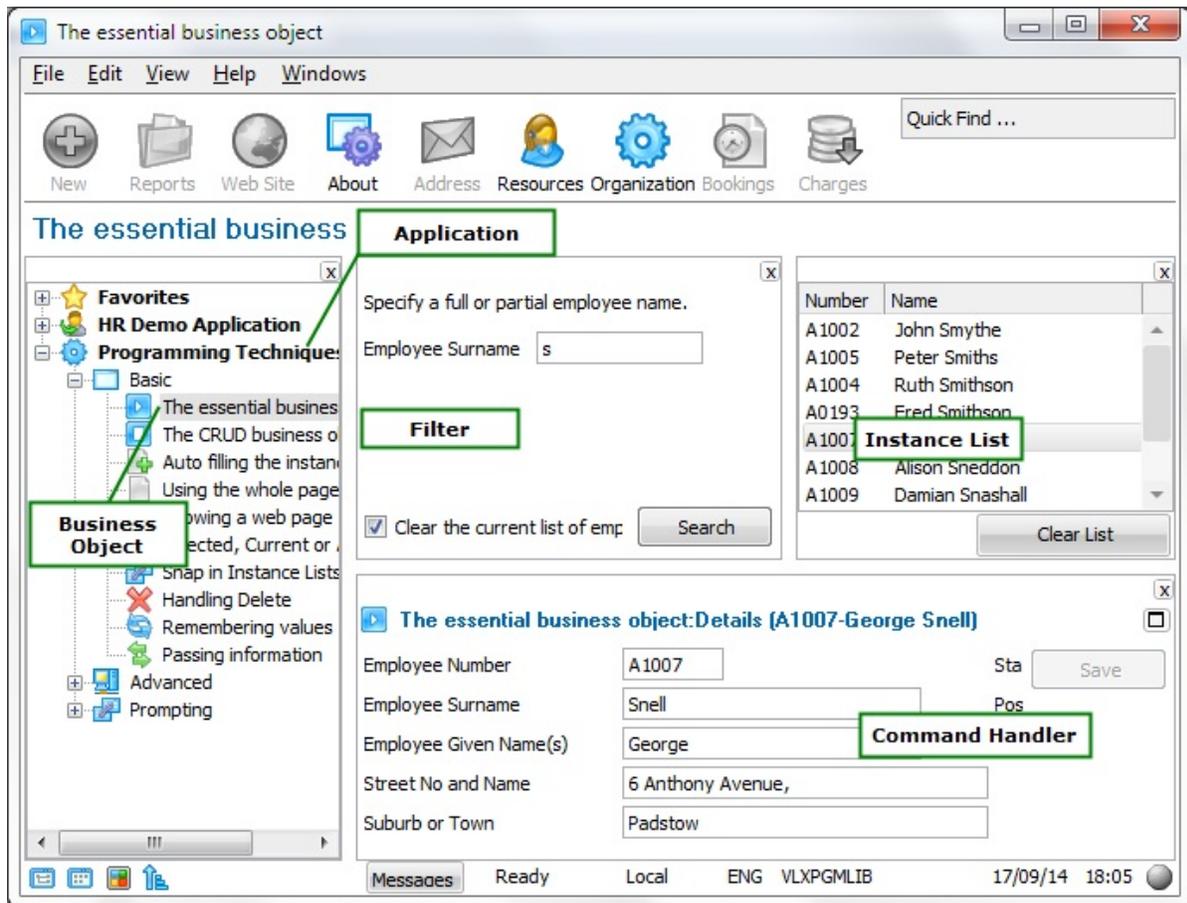
Click Perform All Selected Merges and the merge is performed.

Whenever performing a merge like this, it is recommended that the new combined Framework definition be distributed to all developers.

Use the Framework export facility (see [Export Design](#)) to distribute the Framework definition to every developer.

Key Concepts

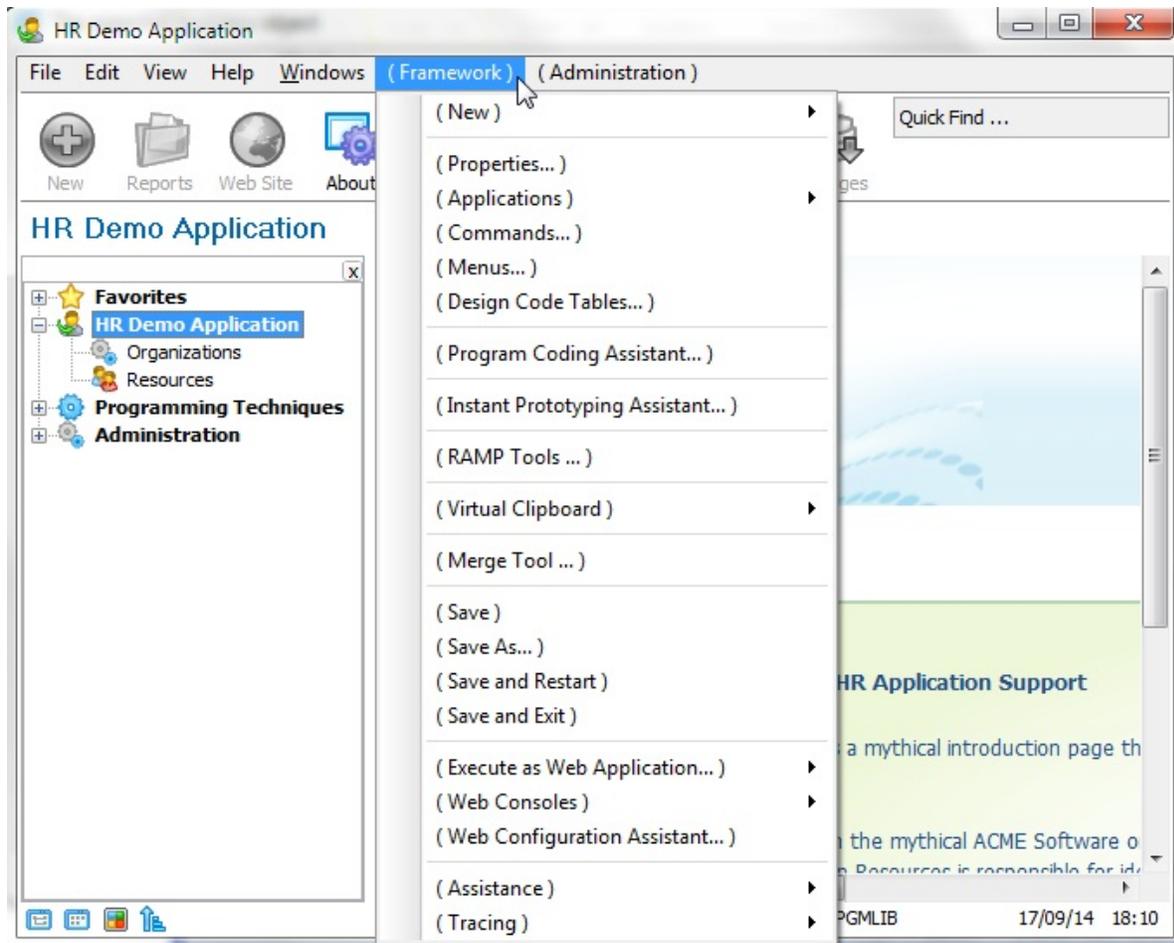
Menu Options in Brackets Application Business Object Instance List Filter
Command Command Handler Images Palette Framework Window



Menu Options in Brackets

Key Concepts

The Framework's Framework and Administration menus and their menu options are in brackets because these menus and options will not be displayed to the end-user when the Framework is executed in User Mode. (Refer to [Starting the Framework](#)).

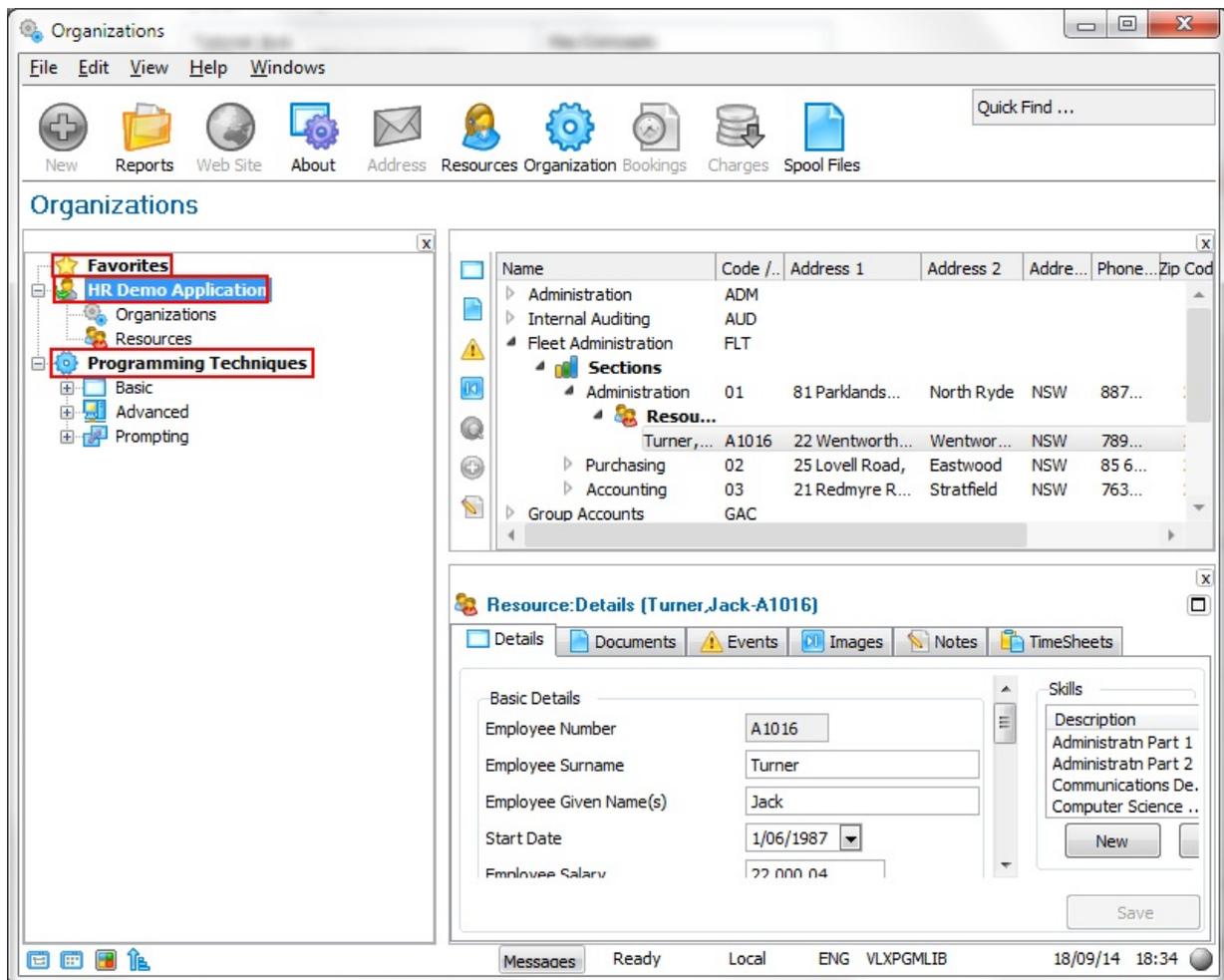


For the same reason, the design-time menu options in the popup menus are in brackets.

Application

Tutorial: [VLF001 - Defining Your HR Application](#) [Key Concepts](#)

An application provides a grouping for [Business Objects](#). There can be several applications in the Framework.



In a commercial environment applications could be, for example, Human Resources, Manufacturing and Payroll.

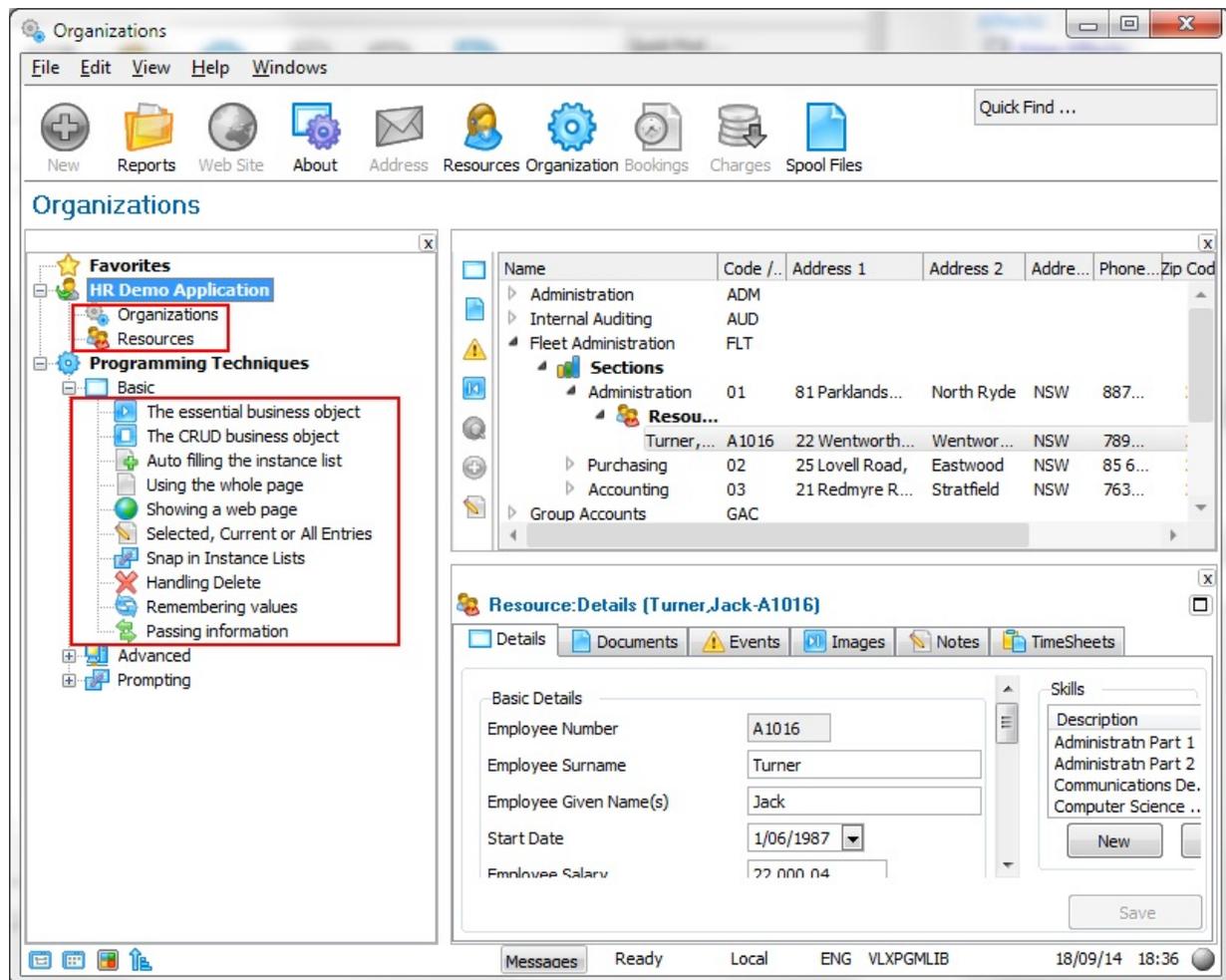
Note: Do not take the word Application too literally. Applications mean different things to different people. In Framework terms an Application just means a logical grouping of business functions. For example, you may have a single "application" called ERP that actually contains seven sub-applications called Stock, Ordering, Pricing, Customers, Suppliers, Invoicing and System Table Maintenance. If you define a single Framework application called ERP it

may become crowded. In Framework terms these seven sub-applications are probably best defined as individual Framework Applications.

Business Object

VLF002 - Defining Your Business Object Key Concepts

Business objects are objects with which the end-user works. They are the core of your application.



A business object is anything that you want it to be, but preferably it is something that an end-user of your application will inherently conceive to be part of your application. For example, an application called Human Resources will be dealing with business objects such as an Employee and Departments.

Do not directly relate a business object to a database table..

Equally, to an end-user, a report is very much an object that they may produce every day in the course of their business. So creating a business object called

Reports, even though it spans many programming "objects" is a perfectly reasonable thing to do.

Please remember Business Objects are objects that end-users recognize and use in the course of their business, not OO programming "objects" as understood by software developers.

Filter

Tutorial [VLF003 - Prototyping Your Filters](#)

Tutorials: [VLF006WIN - Snapping in A Real Windows Filter](#)

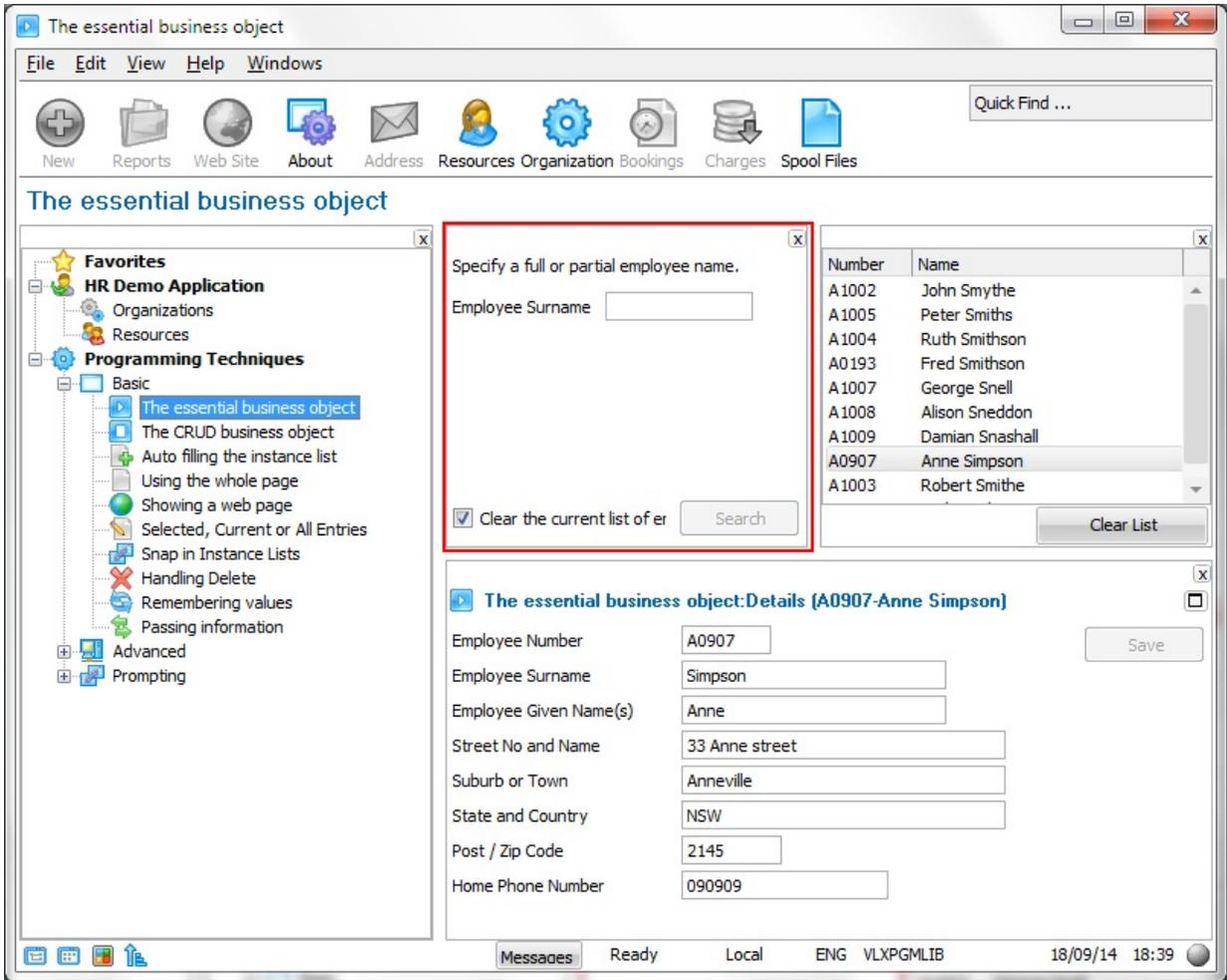
[Key Concepts](#)

[VLF006WAM - Snapping in A Real WAM Web Filter](#)

Filters are used to select business objects and put them into an [Instance List](#). For example, an Employees filter might respond to different user requests by producing an instance lists of:

- all Employees whose names start with SMIT.
- all Employees who have a birthday today.
- all employees that work in the marketing department.
- all employees that started work last year.

The ability to quickly filter information and produce lists is at the heart of many commercial application designs.



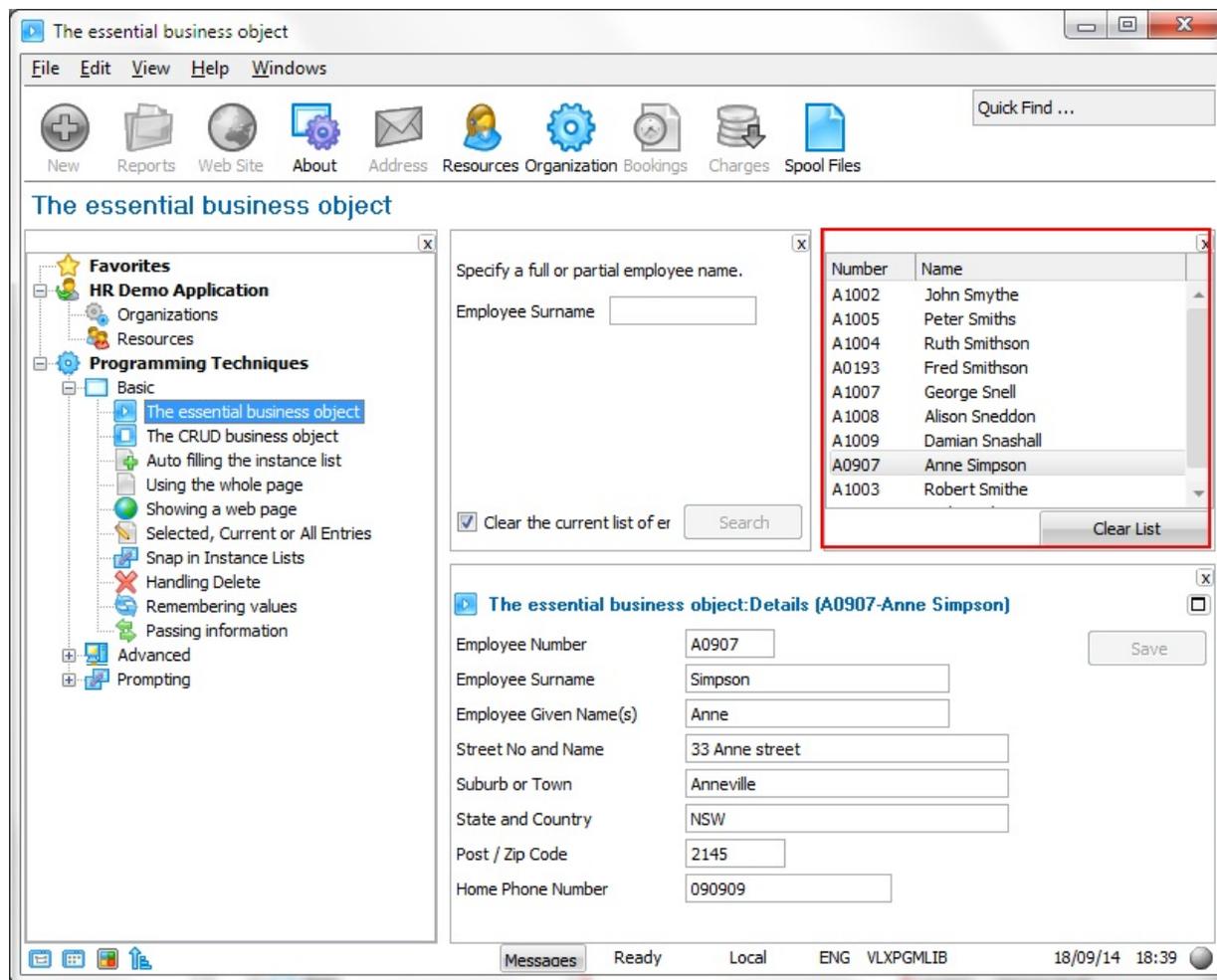
Instance List

Key Concepts

By using a filter an end-user can quickly locate the business objects they want to action (eg: All employees who started work last year).

The Instance List simply displays the list produced by the filter.

Each business object in the instance list is called an instance.



As you can see the filter's job is to filter out employees that match the users selection criteria (in this case a surname that starts with B) and then to feed them into the instance list.

You can customize the settings of an instance list using the properties of the business object.

Command

Key Concepts

You can enable a command for the Framework, an application or a business object and then assign a command handler to it. The command handlers perform the actual processing in the application.

So far we have seen that Filters produce Instance Lists of Business Objects for end-users to action.

For example, a list of all the employees who started work last year might be produced by an Employee business object filter.

To "action" a business object instance the end-user normally executes a command against it.

For example, working with the list of Employees that started work last year an end-user might then choose to execute one of these commands against one or more of the employees displayed:

- Print
- Send Email
- Display History
- Apply a Salary Increase

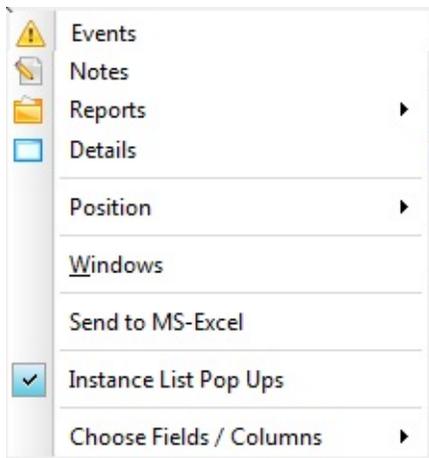
An application designer can actually enable a command for the Framework, an application or a business object and then assign a command handler to it.

The command handlers perform the actual processing in the application (eg: Prints the employee details, Sends the email, Displays the History, etc).

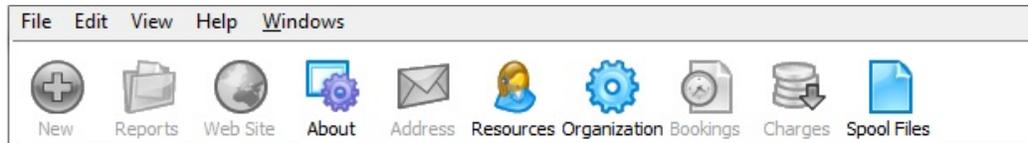
Framework Commands	Global commands, such as Exit, Backup and Help, are available anywhere, anytime.
Application Commands	Application commands are used when you are working within a specific application and they apply to the application as a whole. Backup, Restore and Reports might be valid commands when working with the Human Resources application.
Business Object Commands	Business object (Customers, Products, Orders) commands are used when you are working within a business object and they typically apply to the business object as a complete group or collection. If you are working with a business object named Employee

	then for example New would be a valid command.
Business Object Instance Commands	Business Object Instance commands can only be used when you are working within specific instances of a business object. If you were working with a business object named Employee then Details, Skills, Timesheets, Print and Delete may all be valid commands for a specific employee.

Commands can be visualized by the Framework in several different ways. In a menu on the menu bar or in a pop-up menu:



On the toolbar:



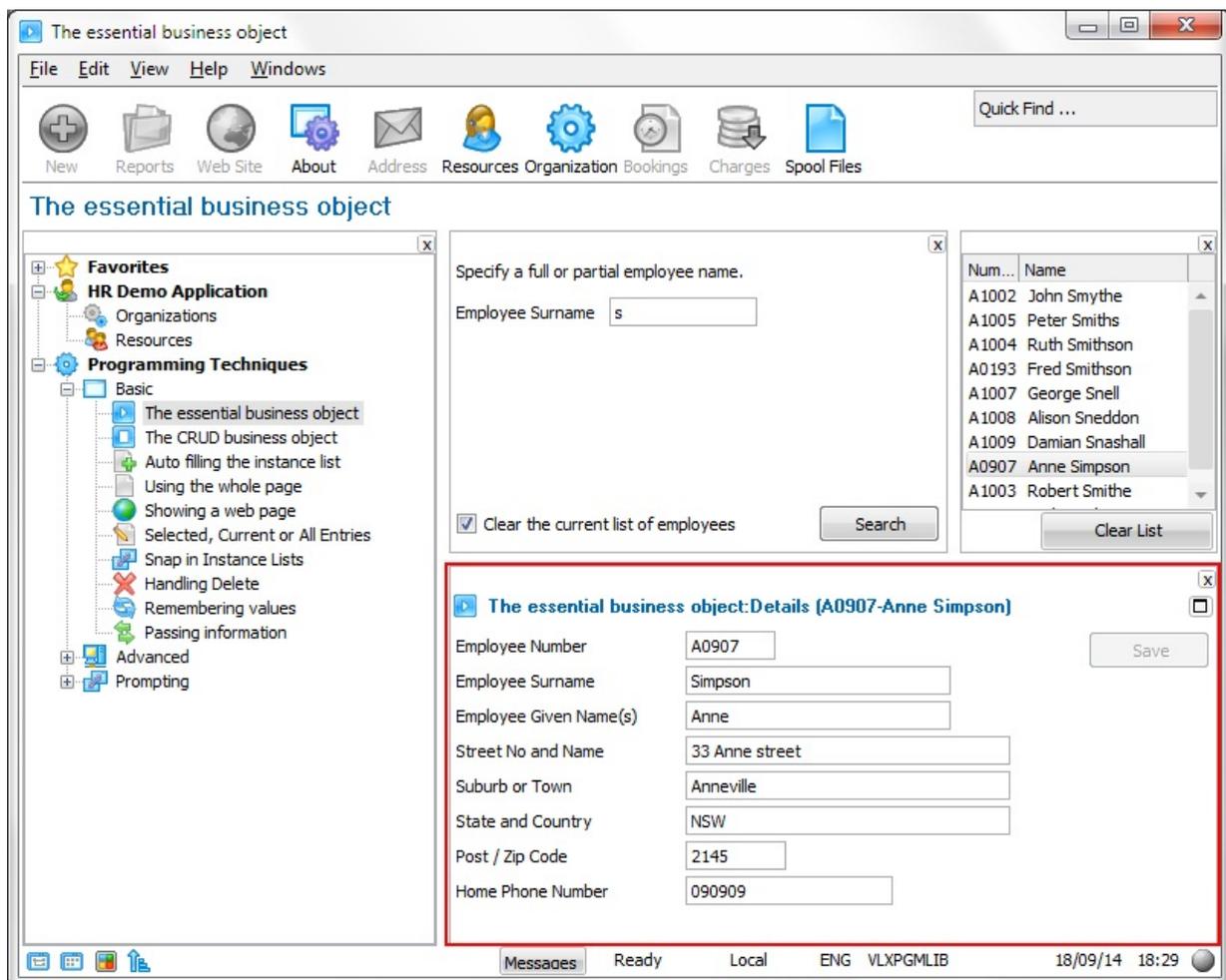
When a user has selected an object and then selects (i.e.: executes) a command then the associated [Command Handler](#) is invoked.

Command Handler

Tutorial [VLF004 - Prototyping Your Commands](#)

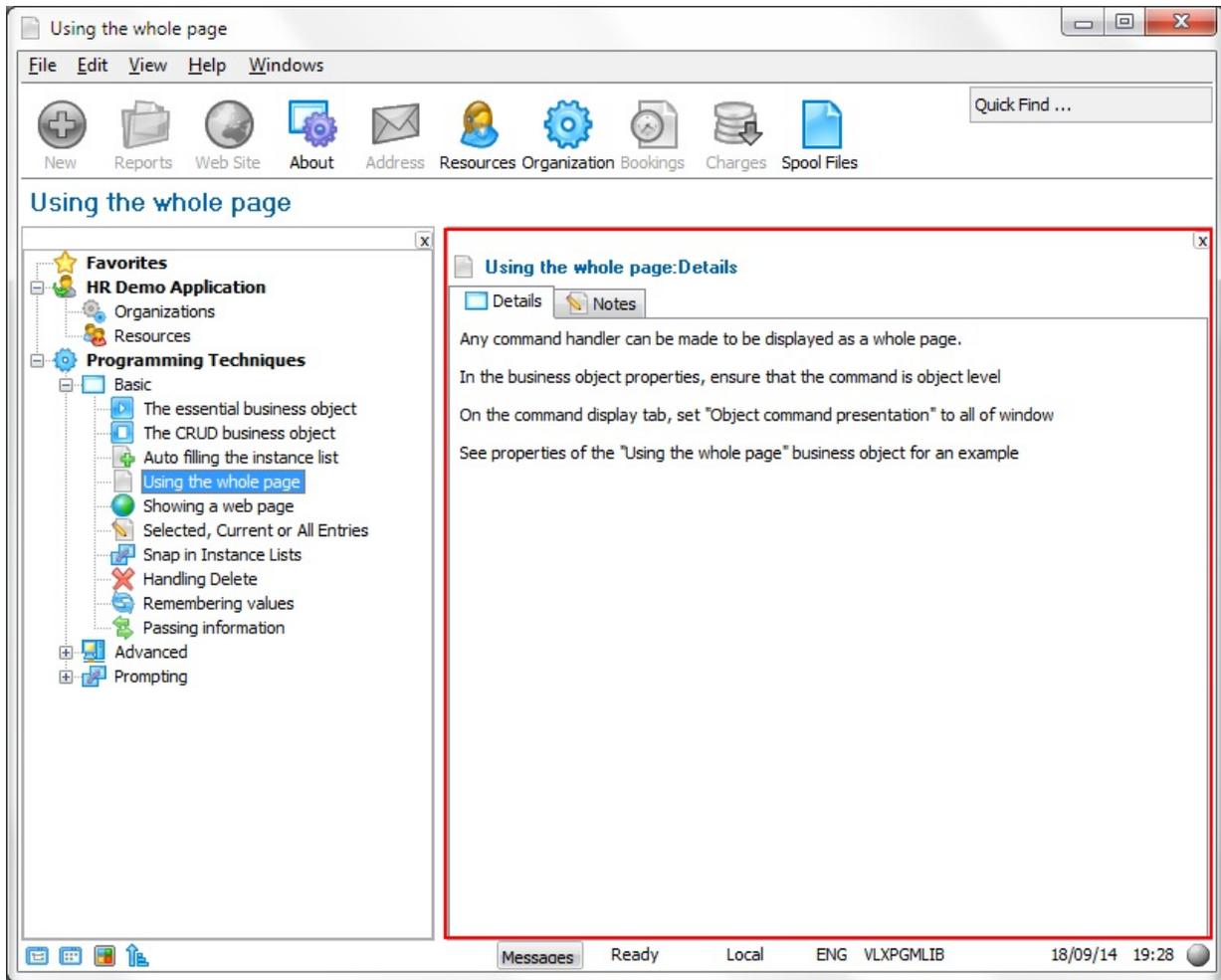
Tutorials: [VLF007WIN - Snapping in a Real Windows Command Handler](#) [Key Concepts](#)
[VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

Command handlers are responsible for executing commands and presenting the results to the user.



In the above picture five command handlers are visible (but only the Details command handler is actually active).

Command handlers can also use up the entire right hand side of the Framework window like this:



Command handlers can also present themselves in a separate window.

Navigation Pane

In the VLF.WIN framework small buttons on the left of the status bar are used to change the view in the navigation pane:



Application designers can still prevent these buttons from being shown using the Framework properties and specify that a specific view is to be used for all end-users.

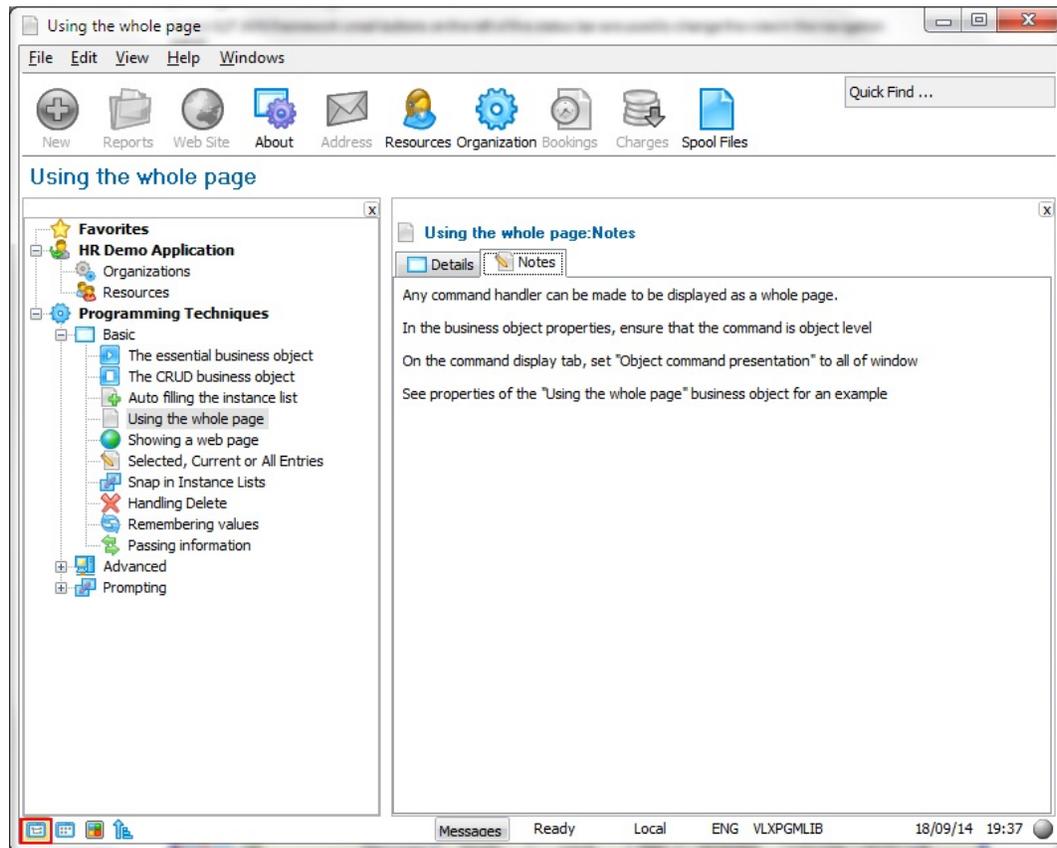
[Tree view](#)

[List view](#)

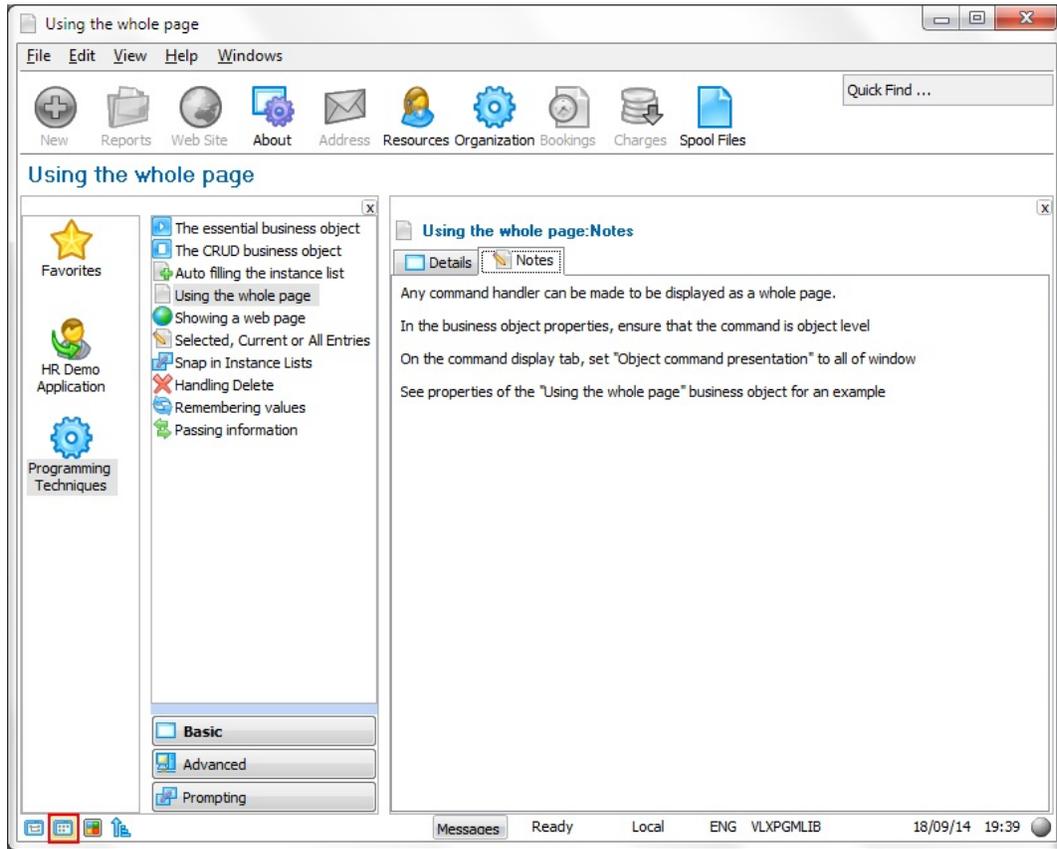
[Drop-down button](#)

[Launching Applications from the Status Bar](#)

Tree view

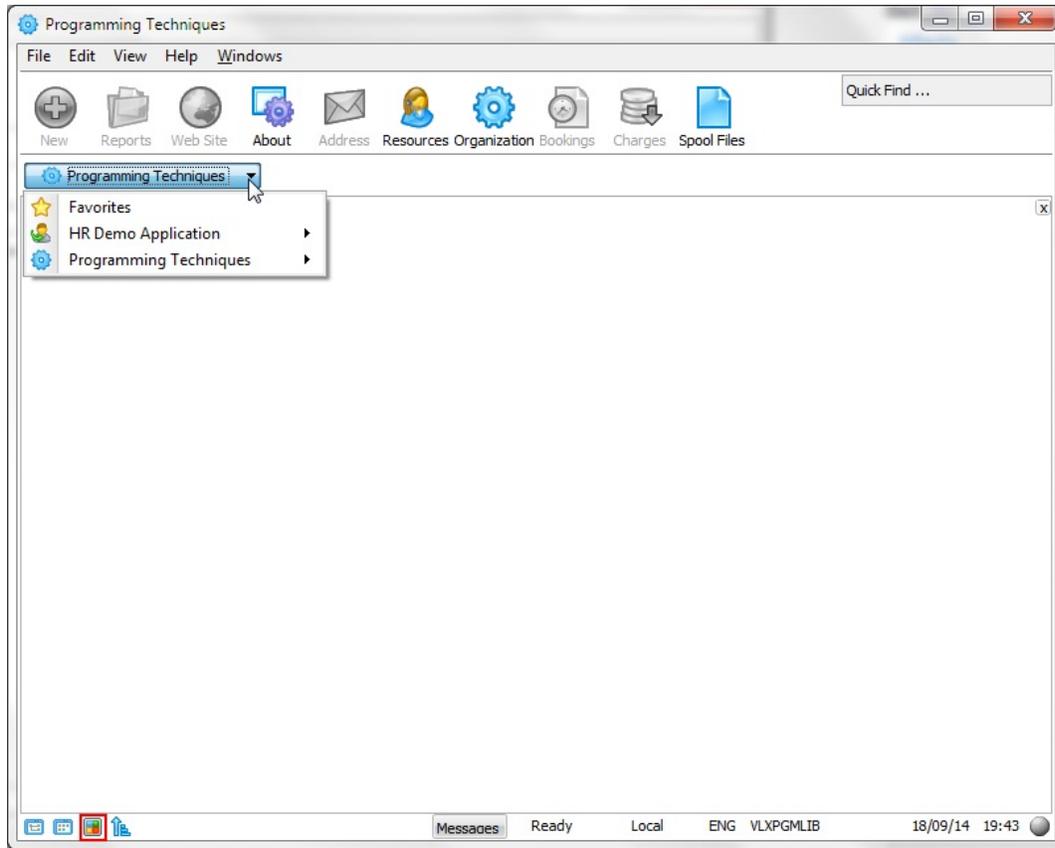


List view



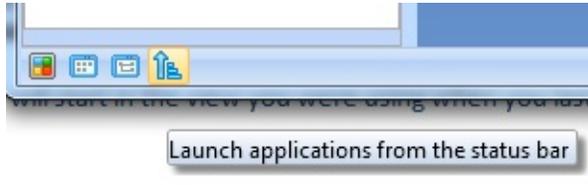
Drop-down button

The drop-down button view hides the navigation pane and gives access to applications and business objects from a drop-down button:

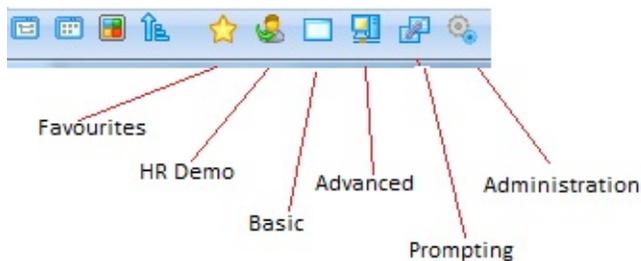


Launching Applications from the Status Bar

When the Framework is executed using RenderType M, the launch button  is displayed in the status bar next to the other navigation pane view buttons:



When the launch button is clicked, applications in the Framework are arranged in the status bar:



If an application has views, the view is visualised. If an application has no business objects it is not shown.

The applications or views respond to two events:

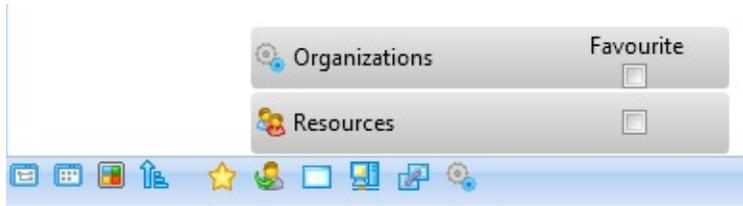
- Mouse hover
- Click

Mouse hover

With a mouse hover a larger icon with the application/view caption appears:



When the larger image is clicked on, the business objects in the application pop up:



If the popup item is clicked, it triggers the default business object's behaviour, as if you clicked on the business object in the navigation pane.

You can also make the business object a favourite by checking the Favourite checkbox.

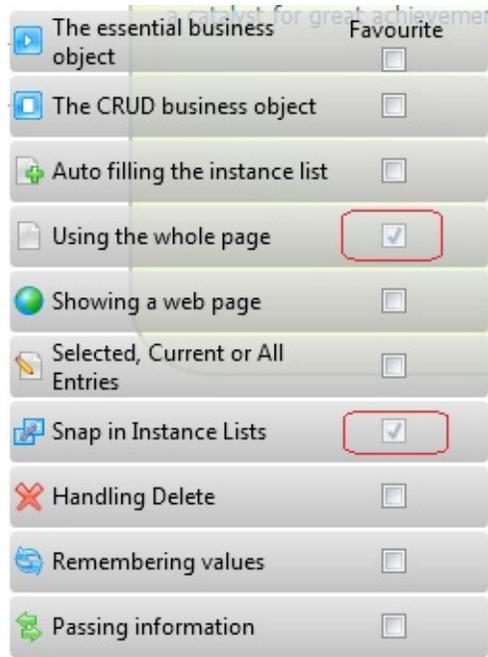
Note: the Business Object will be added to the first application that allows favourites in the sequence they appear. If you need to add it to another application you need to use either the Tree or List Navigation View.

Click

If you click on the application/view, the behaviour is exactly the same as clicking on the larger image.

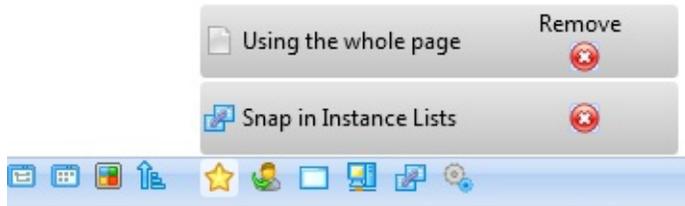
Removing a favourite business object

An application which has business objects that have been made favourites will have the Favorite checkbox is ticked but disabled:



This is because one business object can be a favourite in more than one application.

To remove a business object from a favourite application, hover or click on the favourite application:

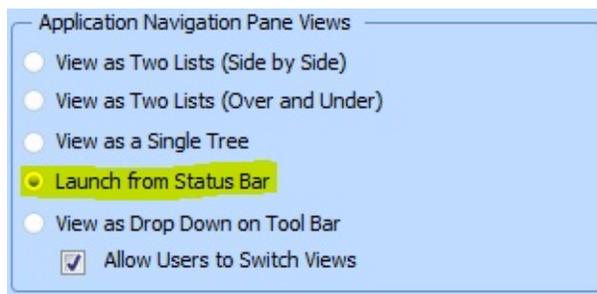


Click the Remove button to remove the business object from the application's Favorites.

Warning: due to space constraints, this navigation option may not be suited to Frameworks with a large number of applications and/or applications with large number of business objects. In those cases use any of the other three navigation pane views.

Enabling the Launch button

The display of the launch button is controlled in the Framework properties:



See the description of [Launch from Status Bar](#).

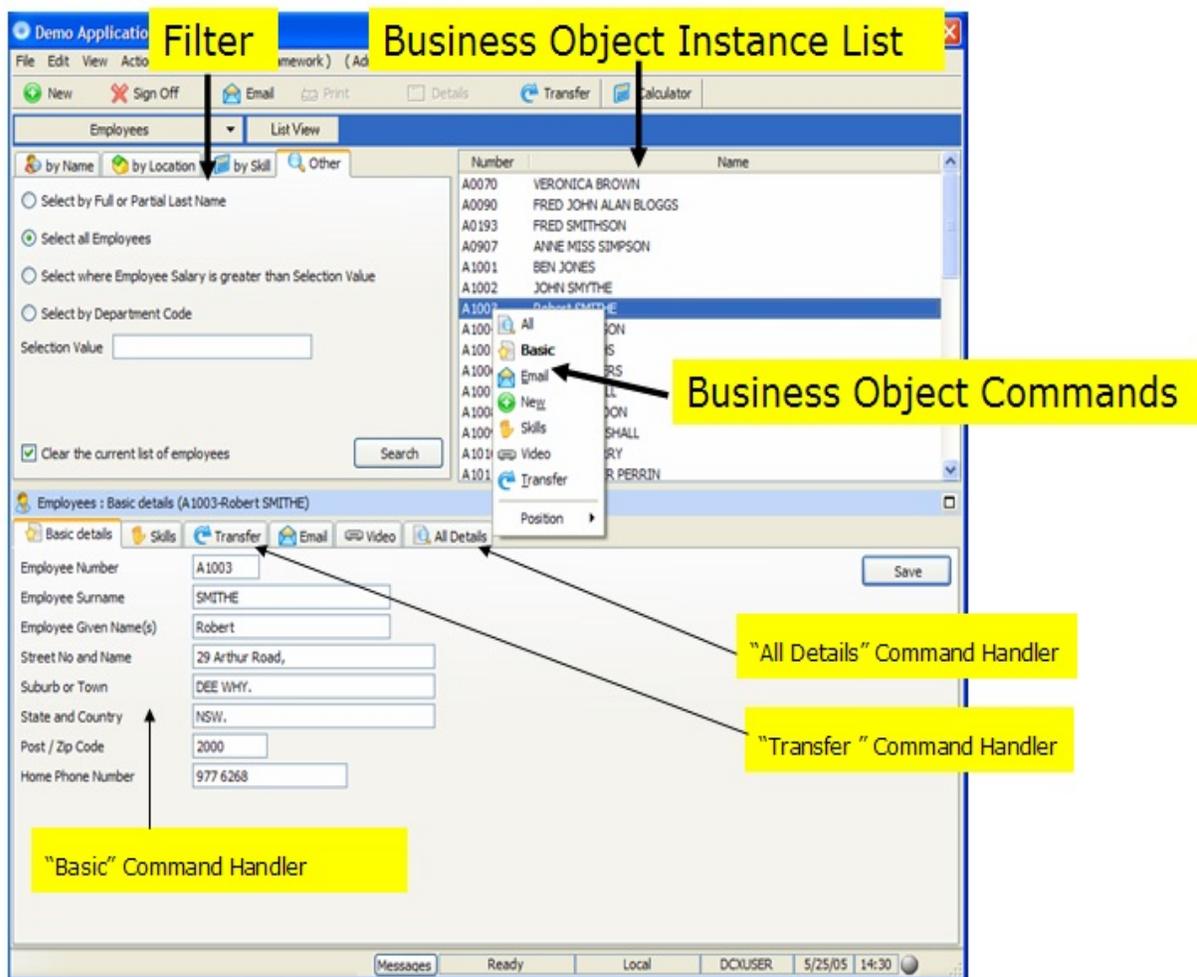
OBJECT-ACTION User Interfaces

If you have used a PC then OBJECT-ACTION should be familiar

If you have used a 5250 then OBJECT-ACTION should be familiar

The preceding sections dealt with the concepts of Business Objects, Filters, Instance Lists, Commands and Command Handlers.

In an "Employees" application, for example, these concepts might be visualized like this:



Here you have a

Filter Where you specify what employees you would like included in your instance list.

Business Object The list of employees that match your filter's search criteria.

Instance
List

Business Object Commands Shown on the pop-up menu when you right click (eg: Basic Details, Transfer, All Details, etc). Sometimes you may execute a command by clicking on an icon on the tool bar.

Command Handlers The various programs invoked when select a command from the pop-up menu.

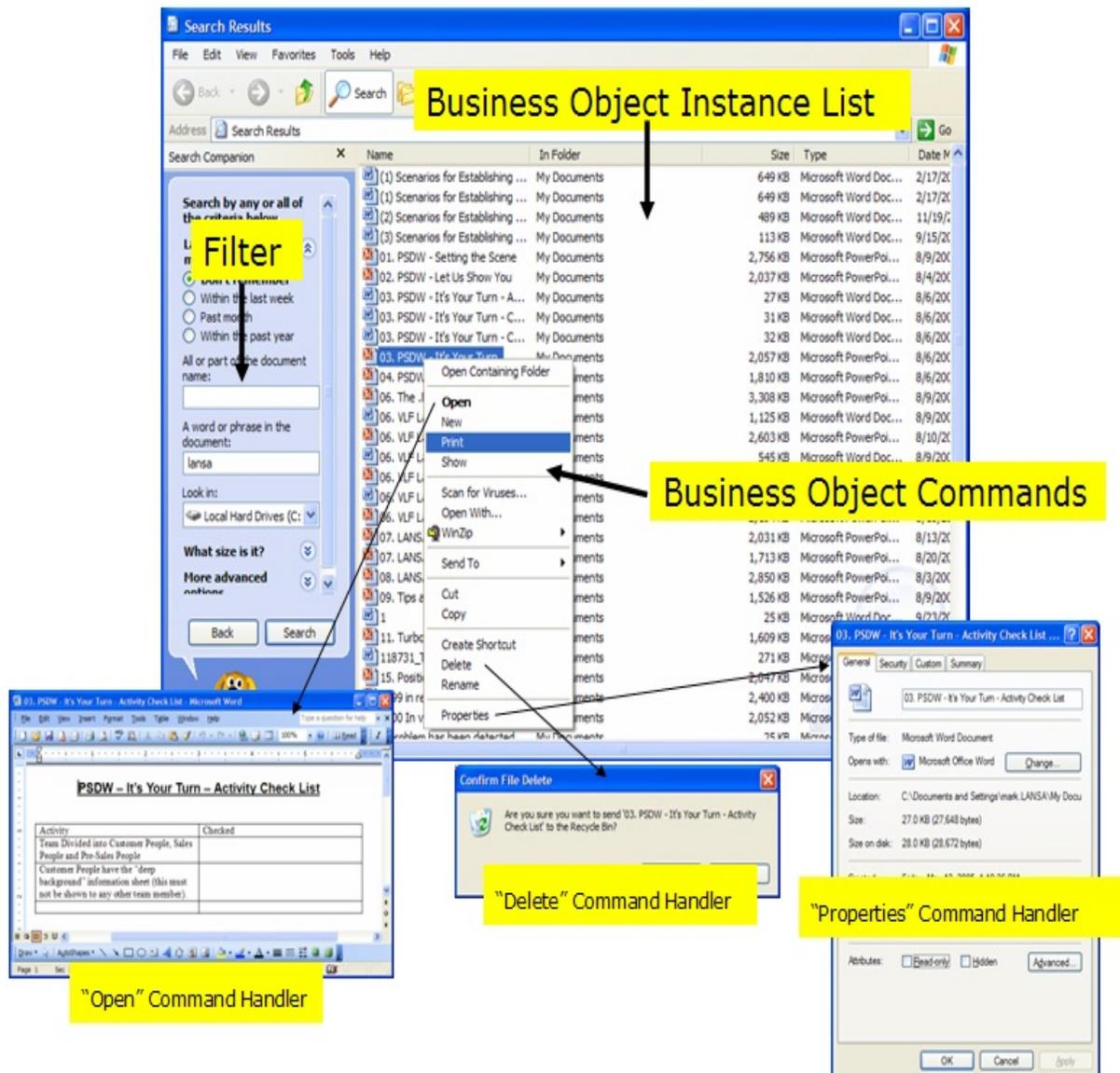
The design principal for these concepts uses an OBJECT-ACTION user interface.

This means that you select an OBJECT then you indicate the ACTION (i.e. Command) that you wish to perform against it.

The OBJECT-ACTION interface is not a Visual LANSa Framework invention.

If you have used a PC then OBJECT-ACTION should be familiar

If you use Start -> Search -> For Files and Folders on your Windows desktop then you end-up with a MS-Windows form that looks like something like this:



Here you also have a:

Filter

Where you specify what documents you would like to search for.

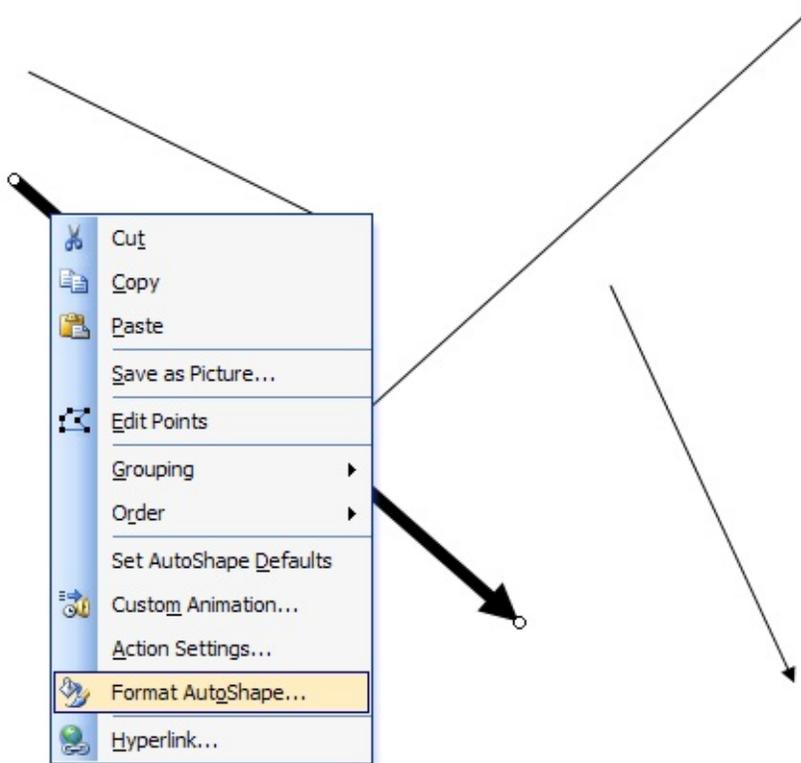
Business Object Instance List

The list of documents MS-Word and MS-PowerPoint documents (say) that match your filter's search criteria. These documents are your business objects.

Business Object Commands Shown on the pop-up menu when you right click (eg: Open, Print, Delete, Properties, etc). Sometimes you may execute a command by clicking on an icon on the tool bar.

Command Handlers The programs that execute when you execute a command. In this example MS-Word handles the "Open" command, a message box confirms that you really want to "Delete" and the document's "Properties" are shown as multiple tabs in a separate form.

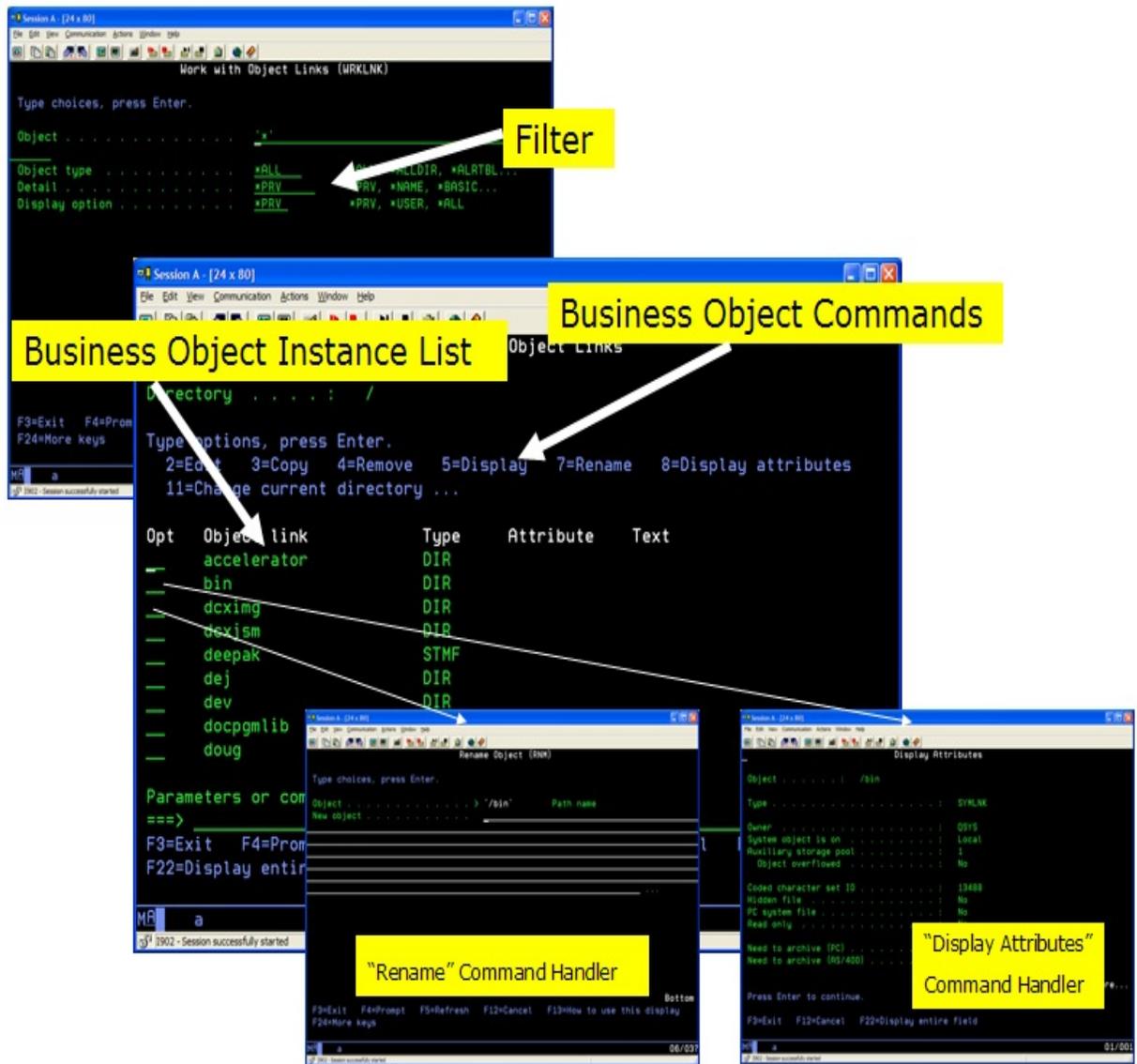
In this example the OBJECT-ACTION approach is being used at a high level. The OBJECT-ACTION design works right down to a very low level as well. Think about how you change the head of an arrow in MS-PowerPoint:



You select the OBJECT (the arrow) then choose the ACTION (Format AutoShape).

If you have used a 5250 then OBJECT-ACTION should be familiar

If you use a classic "Work With XXXX" command on a System i 5250 workstation, such as the WRKLNK (Work with Object Links) command then you should be familiar with 5250 displays like this:



Here you also have a

Filter

Where the WRKLNK command provides you with options to filter the list of links that are displayed. (Many "Work with xxxx"

interfaces allow you to filter inside the main display as well).

**Business
Object
Instance
List**

The list of links that match your filter's search criteria. These links are your business objects.

**Business
Object
Commands**

The Options such as 2=Edit, 7=Rename, 8=Display that you can execute against an individual business object (that is, a link).

**Command
Handlers**

The programs that execute when you execute a command (7=Rename or 8=Display attributes examples are shown).

Again it's the OBJECT-ACTION model underpinning "Work with XXXXX" designs.

You select an OBJECT then you indicate the ACTION (that is, Command) that you wish to perform against it by typing a number beside it (because you can't do right mouse pop-up menus on a 5250 workstation).

You might have even designed commercial "Work with Customers" or "Work with Orders" style applications without even realizing that that the OBJECT-ACTION model you were using was the essentially the same as that used by the MS-Windows desktop and by the Visual LANSA Framework.

Mock Up Filters and Command Handlers

Tutorial [VLF003 - Prototyping Your Filters](#)

Tutorial [VLF004 - Prototyping Your Commands](#)

[Key Concepts](#)

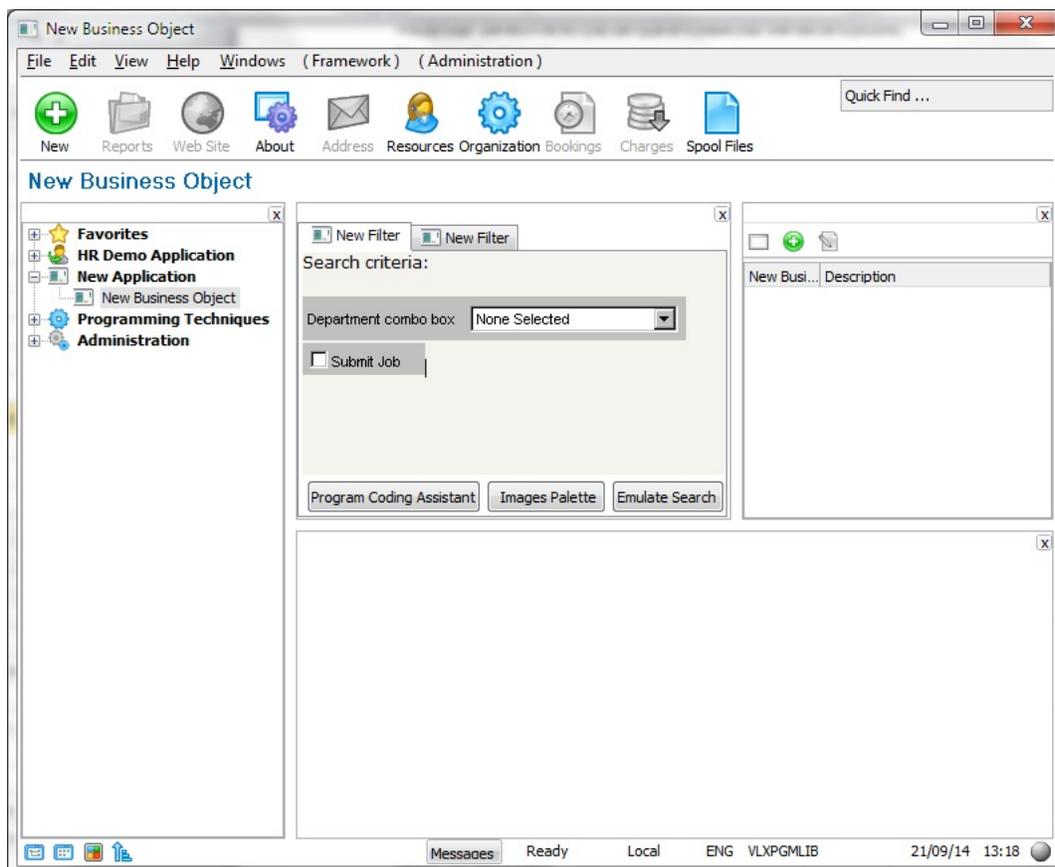
When you are first creating a prototype of your application, you use mock up filters and command handlers that emulate how your application will function. When you turn your prototype into a real application, you snap in your custom made filters and command handlers which provide the actual functionality.

There are two kinds of mock up filters: predefined [Sample Mock Ups](#) and [Mock Up RAD-PAD](#) panels in which you can type and paste your own text and pictures.

Sample Mock Ups

The sample mock up filters and command handlers provide you with a quick way of visualizing what your application will look like. They do not provide any functionality. Their purpose is just to enhance your sense of what your real application will look and feel like.

You cannot change the sample mock ups. This is an example of a sample filter:

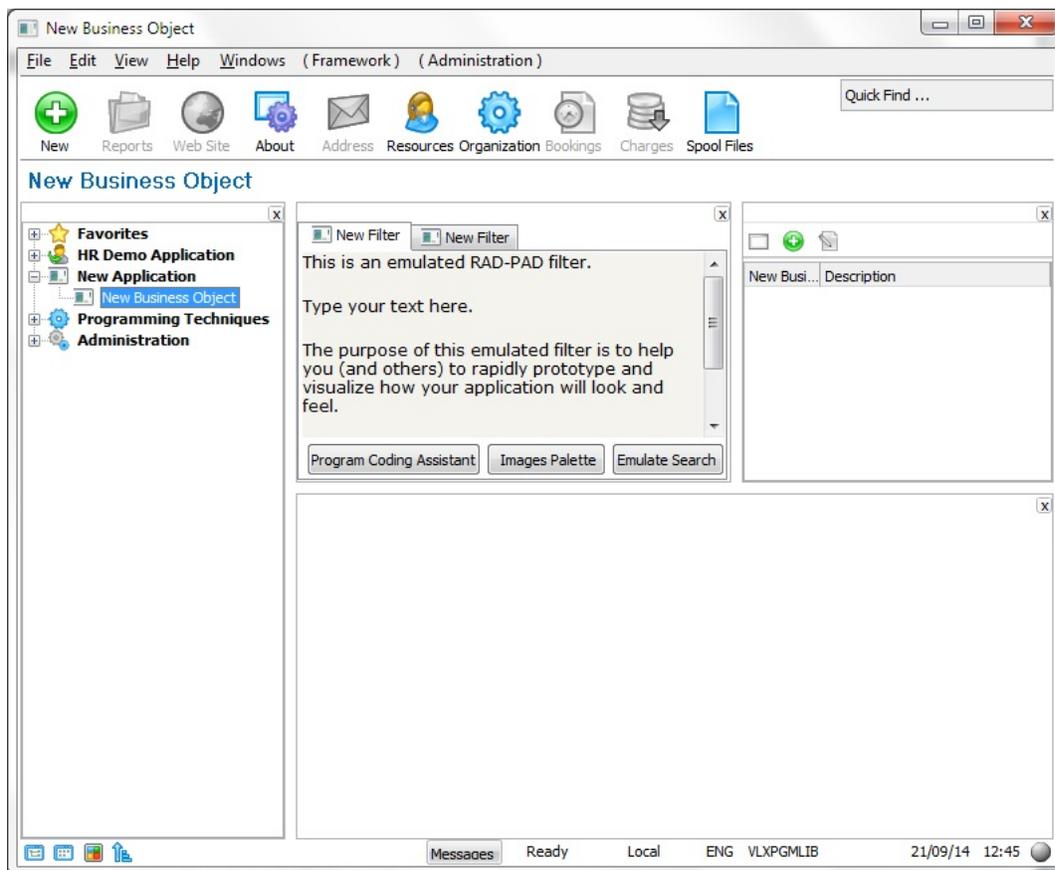


Mock Up RAD-PAD

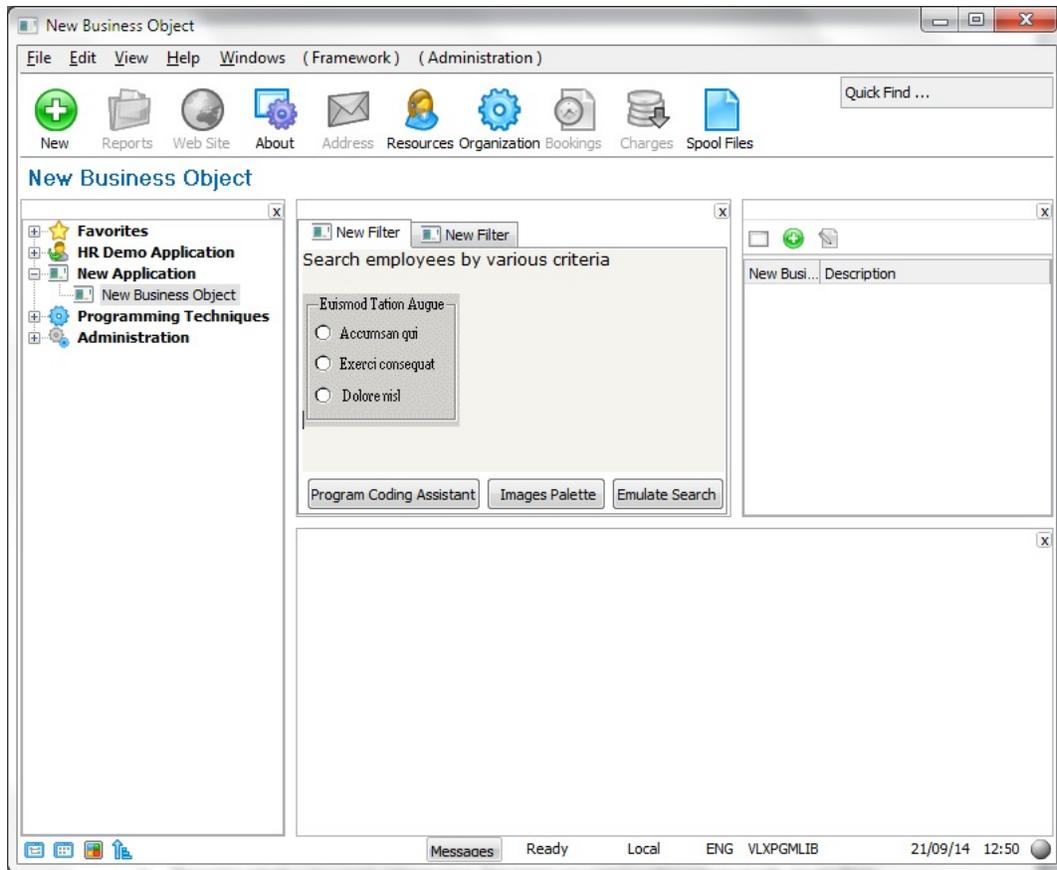
You can use RAD-PAD filters and command handlers to quickly create your own prototype filters and command handlers.

The RAD-PADs are a notepad that allows you to record your design notes and ideas. You can also add simple pictures to your notes by using the [Images Palette](#) to enhance their visualization.

To use a RAD-PAD, select and delete the standard text on the RAD-PAD filter or command handler:



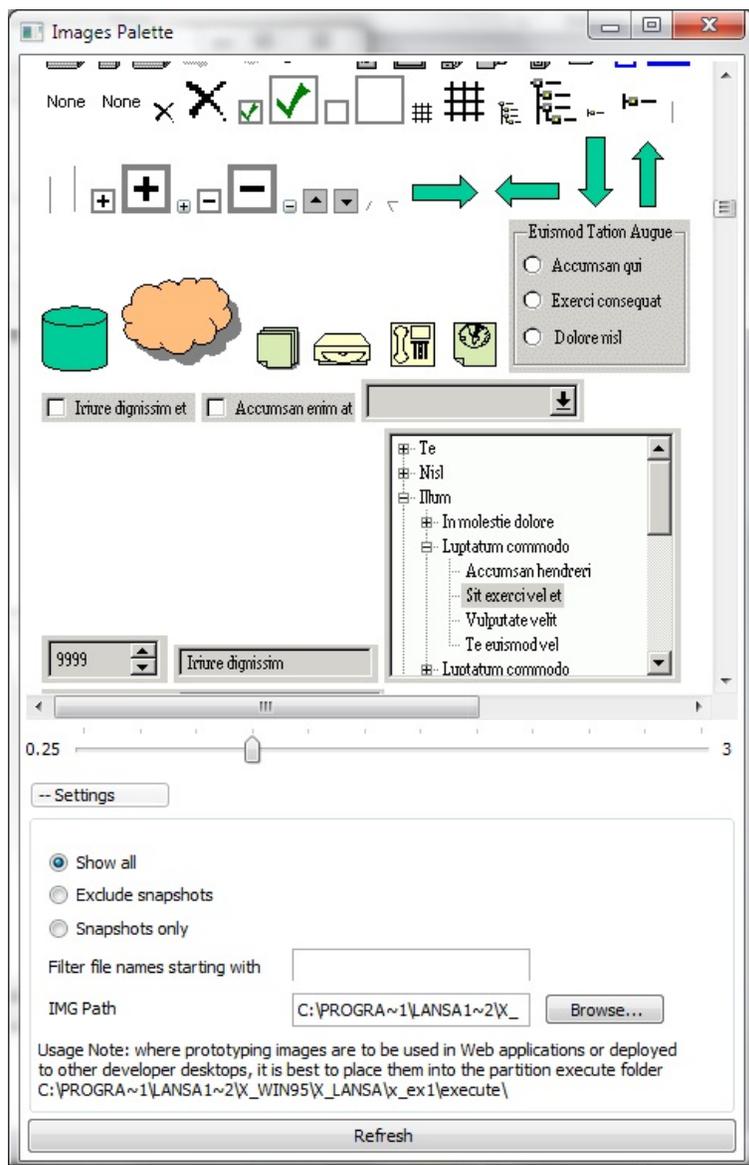
And add your text and pictures:



When creating RAD-PAD filters and command handlers remember:

- They are meant for quick notes with pictures, not for formal screen designs.
- The pad is actually a document containing lines. This means you cannot position pictures exactly. Use the Enter key to add new lines and add blanks or tabs to position items.

Images Palette



Use the Images Palette to quickly add some images to your prototype filters or command handlers:

Click on the Images Palette button to display the palette.

Select the picture you want to insert on your filter or command handler.

Drag and drop or copy and paste the image to your filter or command handler. The palette pictures will paste or drop where the screen input caret (the |) is, not where you point the cursor.

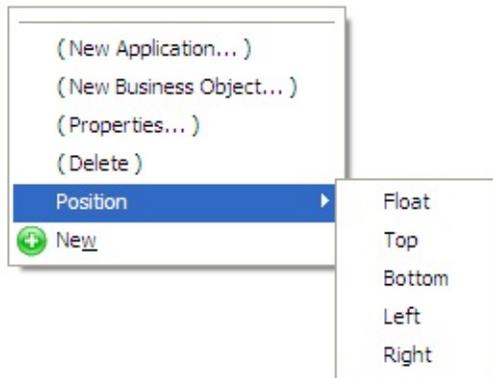
To add your own images to the palette, copy them in .gif format to your partition execute directory. (This is only possible if you use the default value HTML as the [RAD-PAD File Format](#).)

The RAD-PADs are not meant to be used as formal screen designs, only as an aid for quick prototypes. They are a design-time feature not used in the deployed Framework.

Tailoring the Window Layout

The simple Framework layout can be changed by both designers and end-users in Windows.

Use the right button pop-up menu and choose the Position option to rearrange the various components that make up the Framework window into a form that is most suitable for your application:



[Standard Look](#)

[Wider Instance List](#)

[Filter to the Right](#)

[Filter in the Middle with Wider Instance List Area](#)

[Filter and Instance List at the Bottom](#)

[Application Choices to the Right](#)

[Application Choices Moved to Tool Bar](#)

[Standard Layout with Command Handler on Top](#)

[Standard Layout with Command Handler on Top, Filter And Instance List Reversed](#)

[Application Selection on Top](#)

[Application Selection at the Bottom](#)

[Floating Filters](#)

[Floating Instance Lists](#)

Standard Look

Employees

On Tool Bar: Demo Application - 2018, HR, Employees, Departments, Sections, Monthly Reports, Online Reports, Annual Reports, newlook 5250, Web Sites, Programming Techniques

Search Panel: Other, All Views, by Name, by Location, by Skill. Specify a full or partial employee name. Employee Surname: Search - Update, Search - Replace

Number	Name	Phone
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX
A1002	JOHN SMYTHE	047 629 0442
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316
A1004	PAUL SMITHSON	419 5656
A3564	FREDDY BROWN	(02) 567-6758
A0193	FRED SMITHSON	(02) 546-4657
A1007	GEORGE SNELL	764 3562
A1008	ALLAN SNEDDON	476 2198
A1009	DAMIAN SNASHALL	605 8686
A1003	Robert SMITHE	977 6268

Employee: All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details | Skills | Transfer | Email | Video | All Details | Documents

Employee Number: A0090 Save

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-22344XXXX

Commencement Picture:

Messages | Ready | Local | ENG | DCXUSER | 9/20/06 | 12:58

Wider Instance List

Demo Application - 2018

File Edit View Actions Tools Help

New Print Email Details Transfer Calculator

Employees

On Tool Bar

by Name by Location by Skill Other All Views

Specify a full or partial employee name.

Employee Surname

Clear the current list of employees

Search - Update Search - Replace

Number	Name	Phone	Address	Zip
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX	70 MAIN STREET	2220
A1002	JOHN SMYTHE	047 629 0442	,eunevA ytti...	2101
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316	,eunevA en...	2147
A1004	PAUL SMITHSON	419 5656	,daoR mail...	2144
A3564	FREDDY BROWN	(02) 567-6758	121 SMITH STREET	2153
A0193	FRED SMITHSON	(02) 546-4657	evA reltu...	2001

Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-22344XXXX

Save

Commencement Picture



Messages Ready Local ENG DCXUSER 9/20/06 13:00

Filter to the Right

Demo Application - 2018

File Edit View Actions Tools Help

New Print Email Details Transfer Calculator

Employees

On Tool Bar

- Demo Application - 2018
 - HR
 - Employees
 - Departments
 - Sections
 - Monthly Reports
 - Online Reports
 - Annual Reports
 - newlook 5250
 - Web Sites
 - Programming Techniques

Number	Name
A0090	FRED JOHN ALAN BLOGGS
A1002	JOHN SMYTHE
A1005	PETERRRRRRRRRRR SMIT
A1004	PAUL SMITHSON
A3564	FREDDY BROWN
A0193	FRED SMITHSON
A1007	GEORGE SNELL
A1008	ALLAN SNEDDON

by Name by Location by Skill Other All Views

Specify a full or partial employee name.

Employee Surname

Clear the current list of € Search - Update Search - Replace

Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090 Save

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-2234XXXX

Business Phone Number: 654 6475 X432

Commencement Date: 8th MARCH 1992

Employee Salary: 300,000.45

Commencement Picture

Messages Ready Local ENG DCXUSER 9/20/06 13:00

Filter in the Middle with Wider Instance List Area

Demo Application - 2018

File Edit View Actions Tools Help

New Print Email Details Transfer Calculator

Employees

On Tool Bar

- Demo Application - 2018
 - HR
 - Employees
 - Departments
 - Sections
 - Monthly Reports
 - Online Reports
 - Annual Reports
 - newlook 5250
 - Web Sites
 - Programming Techniques

Number	Name	Phone	Address	Zip
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX	70 MAIN STREET	2220
A1002	JOHN SMYTHE	047 629 0442	,eunevA ytti...	2101
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316	,eunevA en...	2147
A1004	PAUL SMITHSON	419 5656	,daoR mail...	2144
A1004	FREDY BROWN	(02) 667 6760	131 SMITH STREET	2153

by Name by Location by Skill Other All Views

Specify a full or partial employee name. Employee Surname

Clear the current list of employees

Search - Update Search - Replace

Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090 Save

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-22344XXXX

Business Phone Number: 654 6475 X432

Commencement Date: 8th MARCH 1992

Commencement Picture



Messages Ready Local ENG DCXUSER 9/20/06 13:01

Filter and Instance List at the Bottom

Demo Application - 2018

File Edit View Actions Tools Help

New Print Email Details Transfer Calculator

Employees

On Tool Bar Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090 Save

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-22344XXXX

Business Phone Number: 654 6475 X432

Commencement Date: 8th MARCH 1992

Employee Salary: 300,000.45

Departments: ADMINISTRATOR DEPT

Commencement Picture

Number	Name	Phone	Address	Zip
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX	70 MAIN STREET	2220
A1002	JOHN SMYTHE	047 629 0442	,eunevA ytti...	2101
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316	,eunevA en...	2147
A1004	PAUL SMITHSON	419 5656	,daoR mail...	2144
A1004	FREDDY BROWN	(02) 667 6760	131 SMITH STREET	2150

by Name by Location by Skill Other All Views

Specify a full or partial employee name. Employee Surname

Clear the current list of employees Search - Update Search - Replace

Messages Ready Local ENG DCXUSER 9/20/06 13:01

Application Choices to the Right

The screenshot shows a software application window titled "Demo Application - 2018". The main area is titled "Employees" and displays a form for editing employee details. The form includes fields for Employee Number (A0090), Surname (BLOGGS), Given Name(s) (FRED JOHN ALAN), Address (70 MAIN STREET, NEWTOWN NSW, AUSTRALIA), and Salary (300,000.45). A "Commencement Picture" is shown as a small portrait of a woman. Below the form is a table listing other employees.

Number	Name	Phone	Address	Zip
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX	70 MAIN STREET	2220
A1002	JOHN SMYTHE	047 629 0442	,eunevA ytti...	2101
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316	,eunevA en...	2147
A1004	PAUL SMITHSON	419 5656	,daoR mail...	2144
A1004	FREDRY BROWN	(02) 667 6768	121 GUMBY STREET	2152

On the right side, there is a tree view titled "On Tool Bar" showing a hierarchy: Demo Application - 2018 > HR > Employees > Departments > Sections > Monthly Reports > Online Reports > Annual Reports > newlook 5250 > Web Sites > Programming Techniques.

At the bottom of the window, there is a status bar showing "Messages", "Ready", "Local", "ENG", "DCXUSER", "9/20/06", and "13:01".

Application Choices Moved to Tool Bar

Demo Application - 2018

File Edit View Actions Tools Help

New Print Email Details Transfer Calculator

Employees List View

Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090 Save

Employee Surname: BLOGGS

Employee Given Name(s): FRED JOHN ALAN

Street No and Name: 70 MAIN STREET

Suburb or Town: NEWTOWN NSW

State and Country: AUSTRALIAaaaaa

Post / Zip Code: 2220

Home Phone Number: 344-22344XXXX

Business Phone Number: 654 6475 X432

Commencement Date: 8th MARCH 1992

Employee Salary: 300,000.45

Departments: ADMINISTRATOR DEPT

Commencement Picture

Number	Name	Phone	Address	Zip
A0090	FRED JOHN ALAN BLOGGS	344-22344XXXX	70 MAIN STREET	2220
A1002	JOHN SMYTHE	047 629 0442	,eunevA ytti...	2101
A1005	PETERRRRRRRRRRR SMITHSSSS...	674 4316	,eunevA en...	2147
A1004	PAUL SMITHSON	419 5656	,daoR mail...	2144
A1004	FREDY BROWN	(02) 667 6766	121 GUMBY STREET	2152

by Name by Location by Skill Other All Views

Specify a full or partial employee name. Employee Surname

Clear the current list of employees Search - Update Search - Replace

Messages Ready Local ENG DCXUSER 9/20/06 13:02

Standard Layout with Command Handler on Top

The screenshot displays a software application window titled "Demo Application - 2018". The interface is organized into several sections:

- Menu Bar:** File, Edit, View, Actions, Tools, Help.
- Command Bar:** New, Print, Email, Details, Transfer, Calculator.
- Section Header:** Employees.
- On Tool Bar:** Employee : All Details (A0090-FRED JOHN ALAN BLOGGS).
- Navigation Tree (Left):** Demo Application - 2018, HR, Employees, Departments, Sections, Monthly Reports, Online Reports, Annual Reports, newlook 5250, Web Sites, Programming Techniques.
- Form Fields (Main):**
 - Employee Number: A0090
 - Employee Surname: BLOGGS
 - Employee Given Name(s): FRED JOHN ALAN
 - Street No and Name: 70 MAIN STREET
 - Suburb or Town: NEWTOWN NSW
 - State and Country: AUSTRALIAaaaaa
 - Post / Zip Code: 2220
 - Home Phone Number: 344-2234XXXX
 - Business Phone Number: 654 6475 X432
 - Commencement Date: 8th MARCH 1992
 - Employee Salary: 300,000.45
 - Departments: ADMINISTRATOR DEPT
- Commencement Picture:** A small portrait photo of a woman.
- Buttons:** Save, Search, Search - Update, Search - Replace.
- List View (Bottom Right):**

Number	Name	F
A0090	FRED JOHN ALAN BLOGGS	3
A1002	JOHN SMYTHE	0
A1005	PETERRRRRRRRRRR SMITHSSSS...	6
A1004	PAUL SMITHSON	4
A3564	FREDDY BROWN	(
A0193	FRED SMITHSON	(
A1007	GEORGE SNELL	7
A1008	ALLAN SNEDDON	4
A1009	DAMIAN SNASHALL	6
- Status Bar (Bottom):** Messages, Ready, Local, ENG, DCXUSER, 9/20/06, 13:02.

Standard Layout with Command Handler on Top, Filter And Instance List Reversed

Employees

Employee: All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090
 Employee Surname: BLOGGS
 Employee Given Name(s): FRED JOHN ALAN
 Street No and Name: 70 MAIN STREET
 Suburb or Town: NEWTOWN NSW
 State and Country: AUSTRALIAaaaaa
 Post / Zip Code: 2220
 Home Phone Number: 344-22344XXXX
 Business Phone Number: 654 6475 X432
 Commencement Date: 8th MARCH 1992
 Employee Salary: 300,000.45
 Departments: ADMINISTRATOR DEPT

Number	Name	Phone	Address
A0090	FRED JOHN ALAN BLO...	344-22344XXXX	70 MAIN S
A 1002	JOHN SMYTHE	047 629 0442	,eune
A 1005	PETERRRRRRRRRRR ...	674 4316	,eune
A 1004	PAUL SMITHSON	419 5656	,dac
A3564	FREDDY BROWN	(02) 567-6758	121 SMITH
A0193	FRED SMITHSON	(02) 546-4657	ev
A 1007	GEORGE SNELL	764 3562	,eun
A 1008	ALLAN SNEDDON	476 2198	,edar

by Name by Location by Skill Other

Specify a full or partial employee name.

Employee Surname

Clear the curre Search - Update Search - Replace

Messages Ready Local ENG DCXUSER 9/20/06 13:03

Application Selection on Top

The screenshot shows a software application window titled "Demo Application - 2018". The interface includes a menu bar (File, Edit, View, Actions, Tools, Help) and a toolbar with icons for New, Print, Email, Details, Transfer, and Calculator. The main content area is titled "Employees" and features a tree view on the left with nodes for HR, Employees, Departments, and Sections. The "Employees" node is selected, displaying a form for "Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)".

The form contains the following fields:

- Employee Number: A0090
- Employee Surname: BLOGGS
- Employee Given Name(s): FRED JOHN ALAN
- Street No and Name: 70 MAIN STREET
- Suburb or Town: NEWTOWN NSW
- State and Country: AUSTRALIAaaaaa
- Post / Zip Code: 2220
- Home Phone Number: 344-22344XXXX
- Business Phone Number: 654 6475 X432
- Commencement Date: 8th MARCH 1992
- Employee Salary: 300,000.45
- Departments: ADMINISTRATOR DEPT

A "Commencement Picture" is displayed on the right side of the form, showing a portrait of a woman. A "Save" button is located in the top right corner of the form area.

At the bottom of the application, there is a table listing employees:

Number	Name	Phone	Address
A0090	FRED JOHN ALAN BLO...	344-22344XXXX	70 MAIN S...
A1002	JOHN SMYTHE	047 629 0442	,eune...
A1005	RETTTTTTTTTTTTTTT	674 4216	...

Below the table, there are search and filter options: "by Name", "by Location", "by Skill", "Other", and "All Views". A search input field is present with the text "Specify a full or partial employee name". A checkbox labeled "Clear the current list of employees" is checked. Search buttons "Search - Update" and "Search - Replace" are also visible.

The Windows taskbar at the bottom shows the system tray with "Messages", "Ready", "Local", "ENG", "DCXUSER", "9/20/06", and "13:04".

Application Selection at the Bottom

Employees

Employee : All Details (A0090-FRED JOHN ALAN BLOGGS)

Basic details Skills Transfer Email Video All Details Documents

Employee Number: A0090
 Employee Surname: BLOGGS
 Employee Given Name(s): FRED JOHN ALAN
 Street No and Name: 70 MAIN STREET
 Suburb or Town: NEWTOWN NSW
 State and Country: AUSTRALIAaaaaa
 Post / Zip Code: 2220
 Home Phone Number: 344-22344XXXX
 Business Phone Number: 654 6475 X432
 Commencement Date: 8th MARCH 1992
 Employee Salary: 300,000.45
 Departments: ADMINISTRATOR DEPT

Commencement Picture

Number	Name	Phone	Address
A0090	FRED JOHN ALAN BLO...	344-22344XXXX	70 MAIN S
A1002	JOHN SMYTHE	047 629 0442	,eune
A1005	PETERRRRRRRRRRR ...	674 4316	,eune
A1004	PAUL SMITHSON	419 5656	,dac
A3564	FREDDY BROWN	(02) 567-6758	121 SMIT

by Name by Location by Skill Other All Views

Specify a full or partial employee name.

Employee Surname

Clear the current list of employees

Search - Update Search - Replace

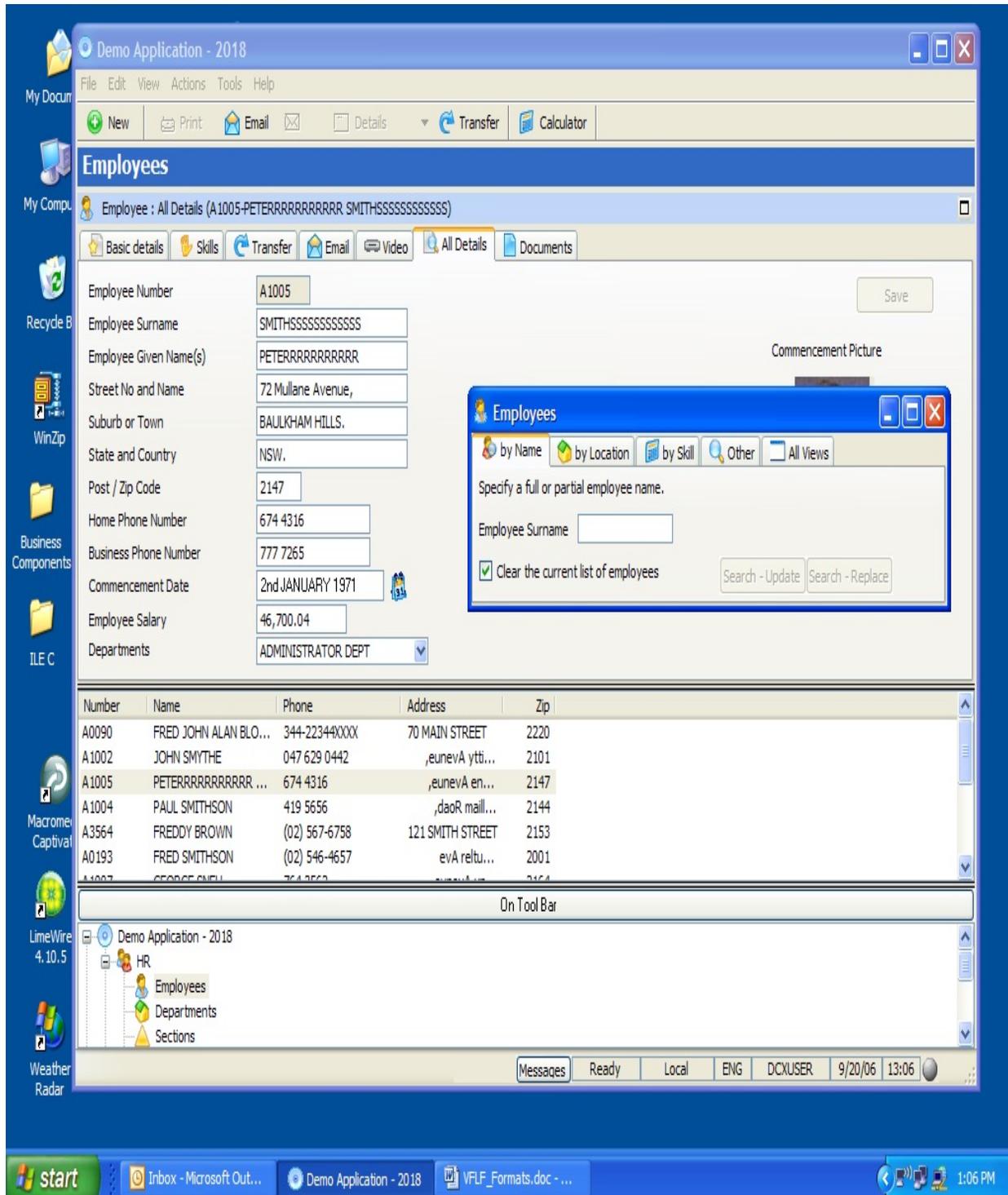
On Tool Bar

Demo Application - 2018

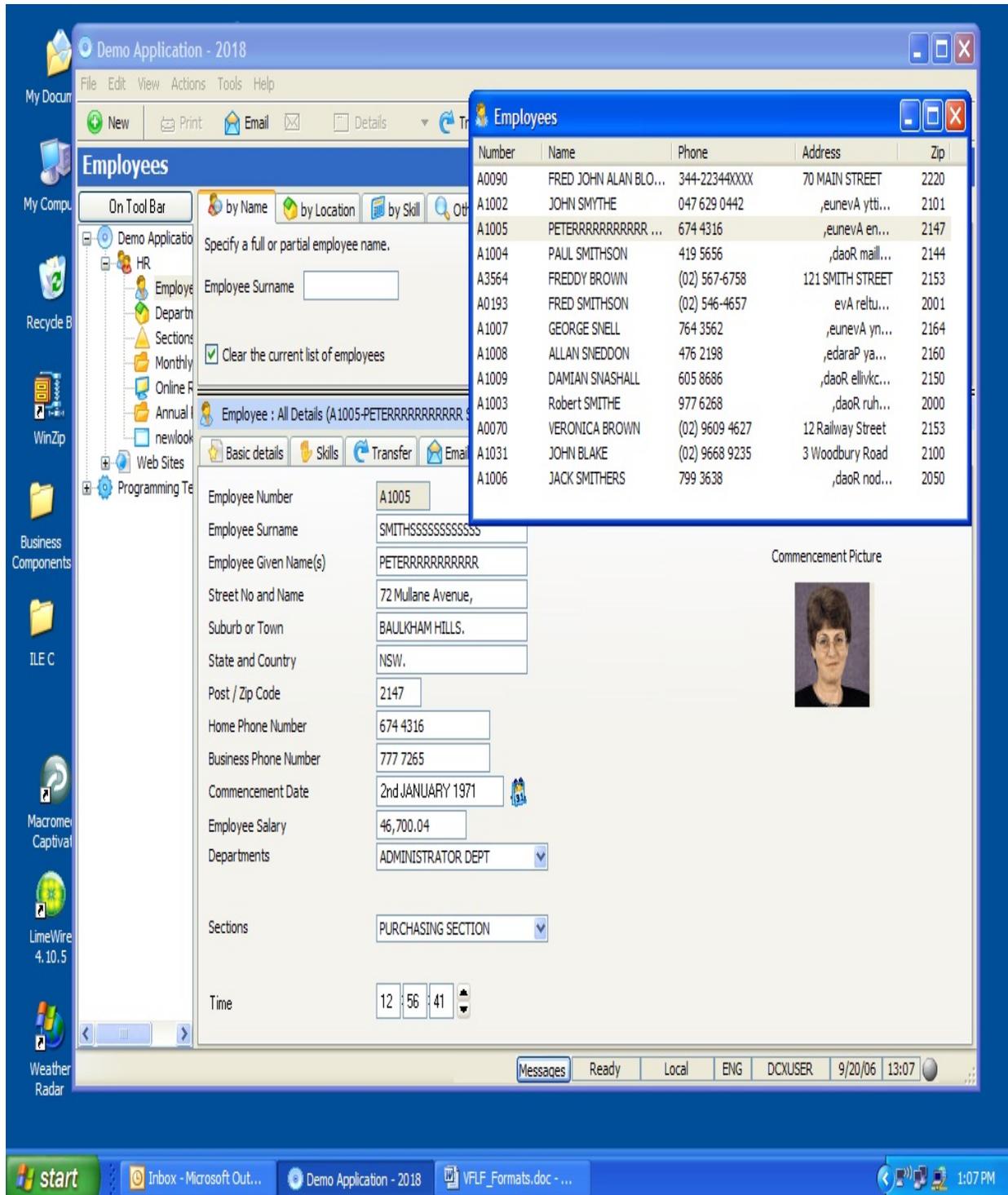
- HR
 - Employees**
 - Departments
 - Sections

Messages Ready Local ENG DCXUSER 9/20/06 13:04

Floating Filters



Floating Instance Lists

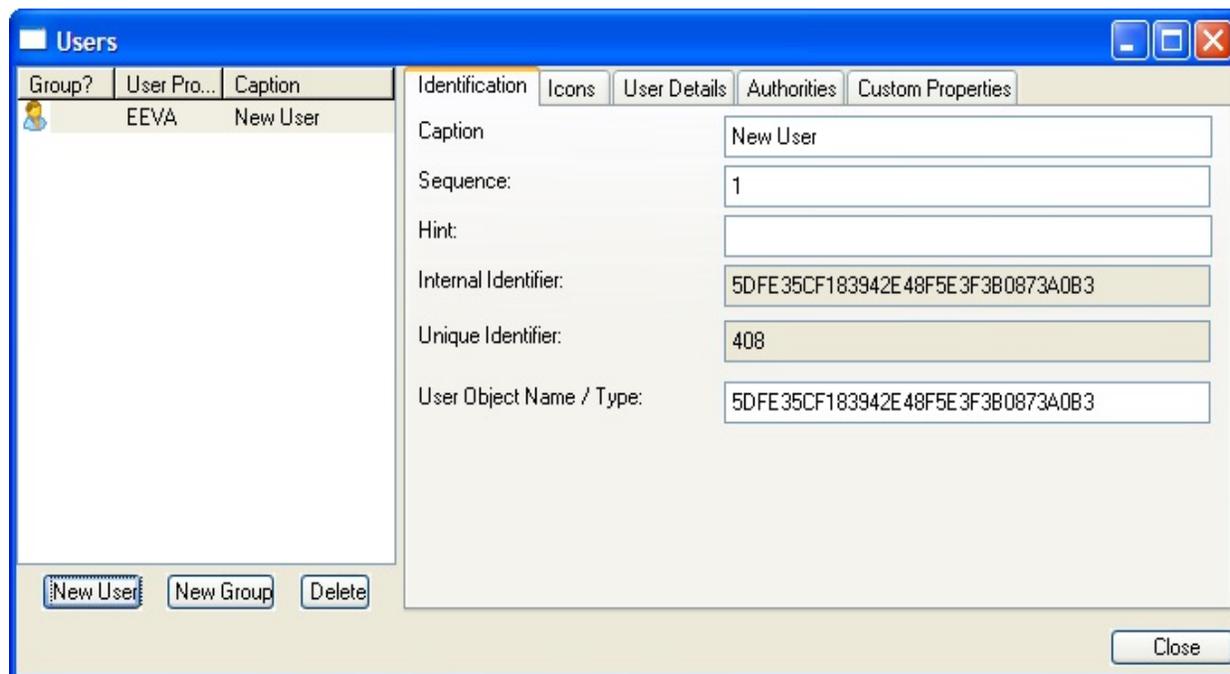


Administration Objects

Use the Administration menu to work with [Users](#) and [Servers](#) defined in the Framework.

Users

This dialog box shows the users registered to use the Framework. It provides an optional facility to help you define and manage your application's user profile definitions.

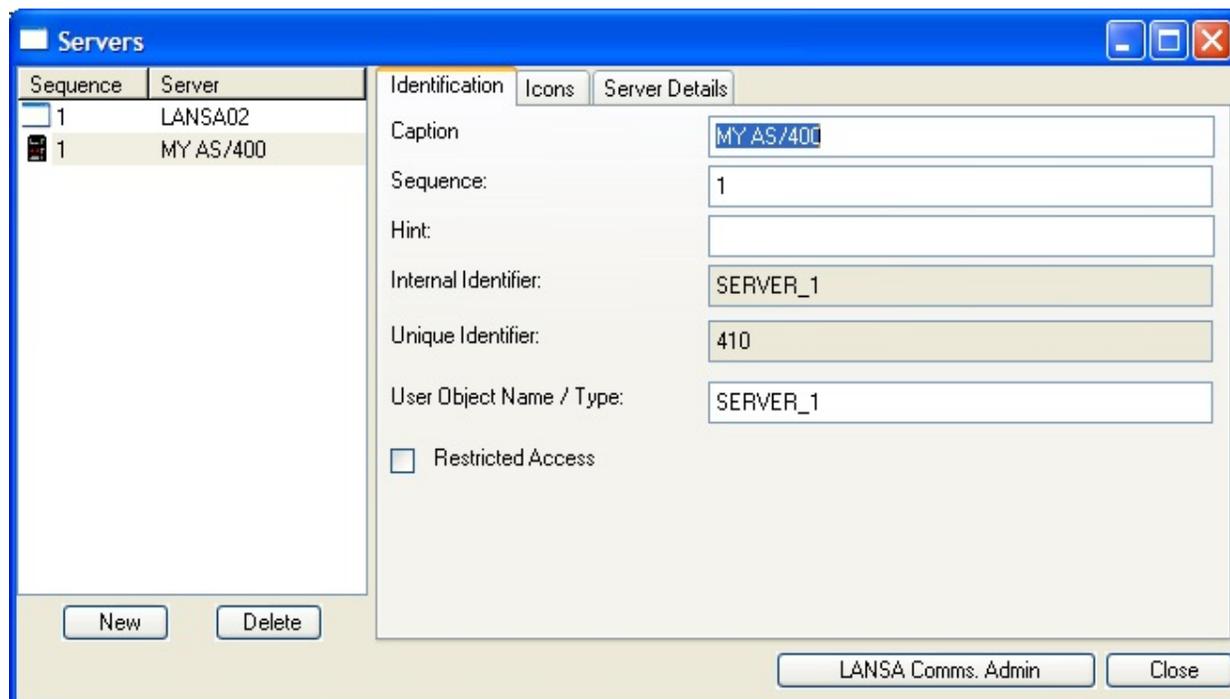


To switch on Framework authority checking, go to the Framework menu, choose the Properties... option and select the Framework Details tab. On this tab, the "Users, Authority and sign on settings" panel is where Framework authority is configured. For details, refer to [Framework Details](#).

For more information, see [Users, Groups and Security](#).

Servers

This dialog box shows the servers defined in the Framework. It provides an optional facility to help you define and manage your application's server definitions.



The servers defined here will be displayed in the Connect dialog when the user logs on to the Framework.

A user with authority to use the Administration menu can add servers and work with their details.

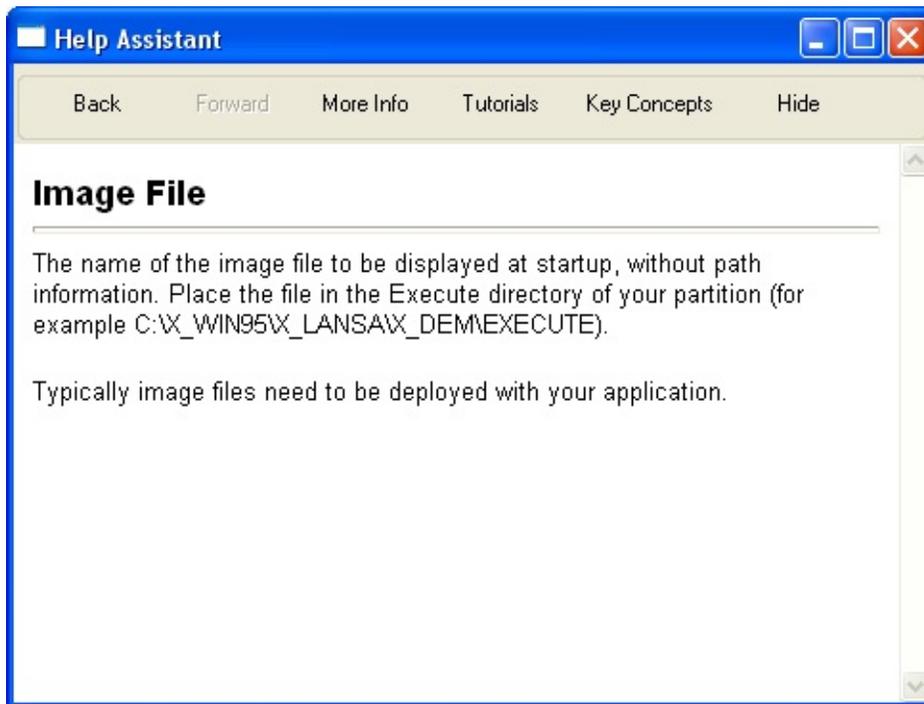
Server-related options are also defined in [Framework Details](#) dialog box.

For more information see [Server Profile Management and Issues](#).

Help and Tutorials

The Visual LANSA Framework has a help assistant and tutorials.

The Help Assistant provides help when you are developing applications in the Framework. Either leave the Assistant window open or close it and press F2 when you want help for an object.



The [Tutorials](#) provide you with simple step by step instructions of how to create a Framework application.

Visual LANSA Framework Guide

Hide Previous Next Back Forward Refresh Home Print

Contents Search

- Visual LANSA Framework Guide
 - Introduction
 - What's New
 - Getting Started
 - Key Concepts
 - Building the Application
 - Tutorials**
 - Before You Use the Tutorials
 - Tutorials for Prototyping
 - Tutorials for Windows Applications
 - Tutorials for Webevent Web Browser A
 - Tutorials for WAM Web Browser Applic
 - Tutorials for Deployment
 - Optional Tutorials
 - Frequently Asked Questions
 - Framework Programming
 - Fast Parts
 - Advanced Topics
 - Troubleshooting
 - Application Performance
 - Definitions
 - Appendix

Back Forward Search Contents SET

Tutorials

The Visual LANSA Framework Tutorials are a set of exercises designed to introduce and reinforce the fundamental skills required to build applications with the Framework.

The tutorials guide you in creating a sample Human Resources application.

Prototype

- [VLF000 - Execute Framework Application](#)
- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)

Windows Applications

- [VLF006WIN - Snapping in A Real Windows Filter](#)
- [VLF007WIN - Snapping in A Real Windows Command Handler](#)

Webevent Web Browser Applications

- [VLF006WEB - Snapping in A Real Webevent Filter](#)
- [VLF007WEB - Snapping in a Real Webevent Command Handler](#)

WAM Web Browser Applications

- [VLF006WAM - Snapping in A Real WAM Web Filter](#)
- [VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

Deployment

- [VLF008WIN - Deploying the Windows Framework](#)

Optional Tutorials

- [VLF009WIN - Adding Instance List Columns in Windows Applications](#)
- [VLF009WEB - Adding Instance List Columns in Web Applications](#)

See Before You Use the Tutorials

Building the Application

These are the steps you perform to create your application:

Before building your application, you would usually [Personalize Your Framework](#) to make it reflect the company you are building the application for.

Prototype:

[Define Your Application](#)

[Define Your Business Objects](#)

[Optionally Group Business Objects into Application Views](#)

[Prototype Your Filters](#)

[Prototype Your Commands and Their Handlers](#)

[Validate Your Design](#)

Implement:

[Create Your Own Filters](#)

[Create Your Own Command Handlers](#)

[Optionally Create Your Own Instance List](#)

Deploy:

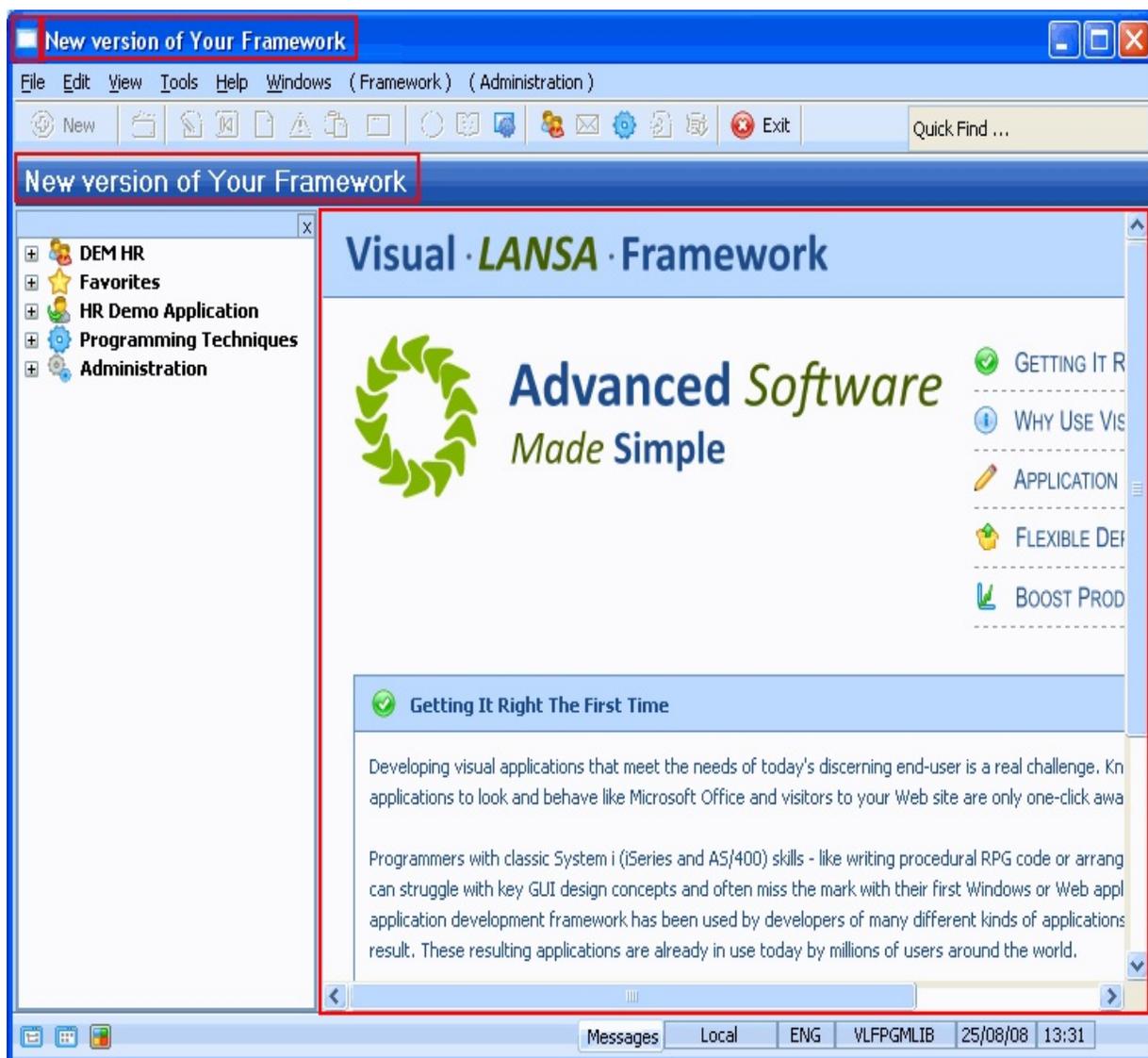
[Deploy the Application](#)

Personalize Your Framework

There are some simple things you should do to make your Framework reflect the company you are building the application for. Even if you are just doing a proof of concept, it is important to make the Framework appear to be the customers software, not LANSA's.

Note that you can also change the appearance of the Framework by [Tailoring the Window Layout](#) .

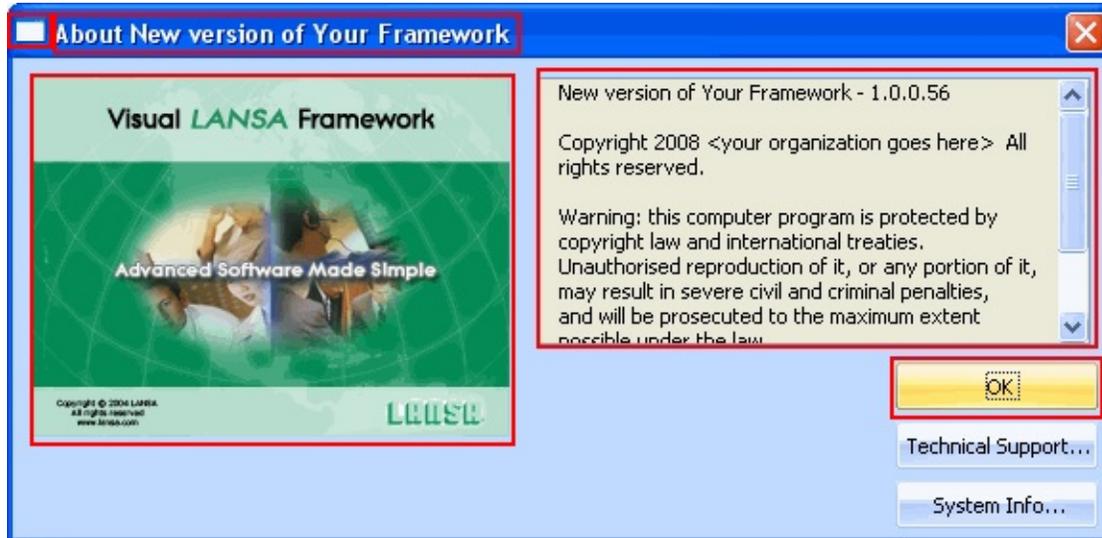
As shipped, when the Framework is run, it looks like this:



The shipped splash screen and the link to the web page are only intended as

placeholders. They should be changed to reflect the company the Framework is being built for. So should the Framework caption and icon.

When the user chooses the Help menu option About Framework in the shipped version of the Framework they see this:



The image, the text, and the url that the technical support button links to (www.lansa.com) should all be changed.

An image can also be added to the top of the sign-on screen:



To Change these Settings

Go to Framework properties to change most of these settings:

[Framework Caption](#)

[Framework Icon](#)

[Framework splash image](#)

[The technical support URL](#)

[The Sign on Screen image](#)

If you want to make a change to [First Splash Screen](#), you need to make a change to your entry point form.

Framework Caption

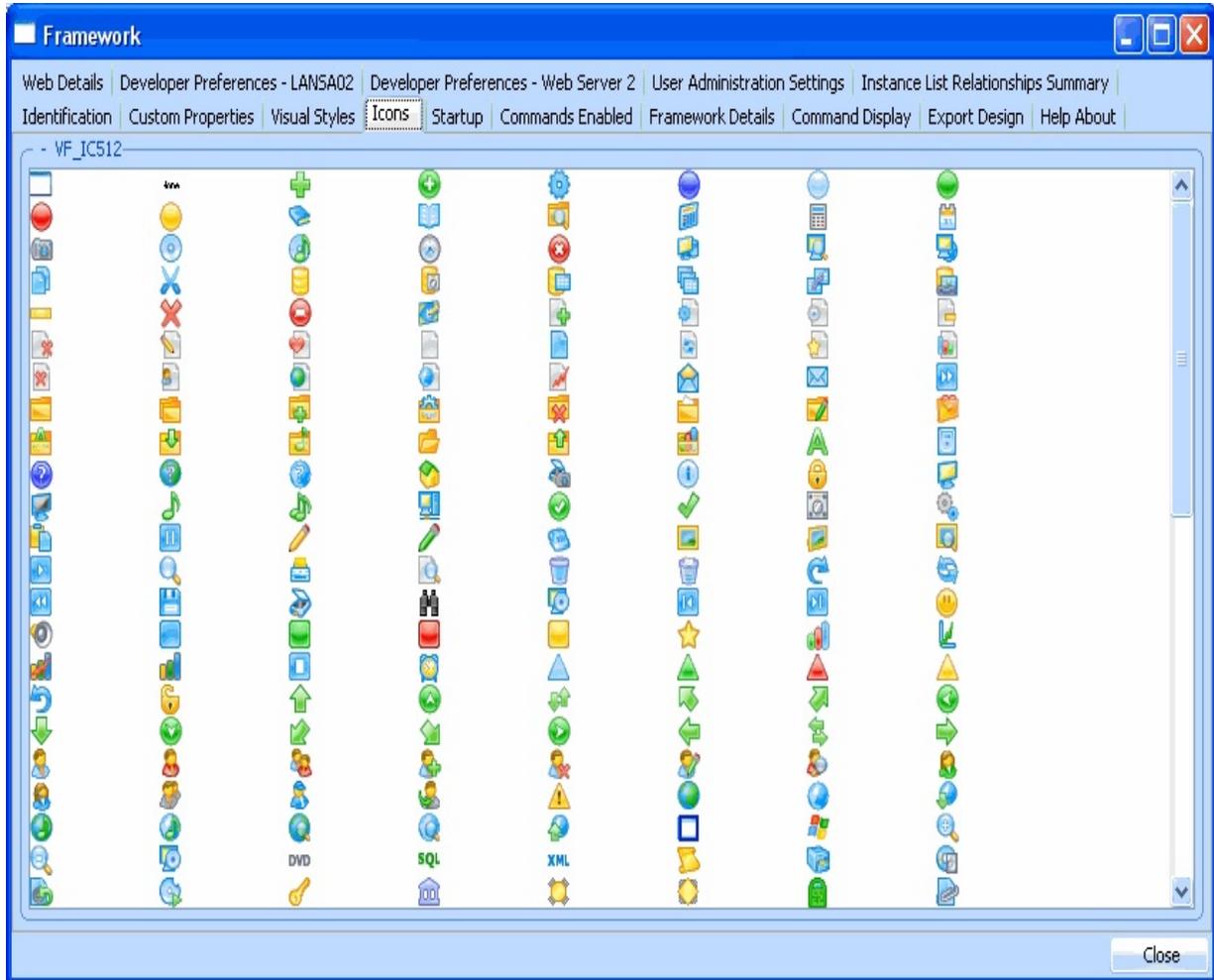
Use the Caption property on the Identification tab:

The screenshot shows a window titled "Framework" with a tabbed interface. The "Identification" tab is selected and highlighted with a red border. The "Caption" property is set to "New version of Your Framework" and is also highlighted with a red border. Other properties include "Hint" (empty), "Sequence" (1), "Internal Identifier" (VF_SHIPPED), "Unique Identifier" (1), and "User Object Name / Type" (SHIPPED_FRAMEWORK). There are several checkboxes for window behavior, a dropdown for "Allow this Object to be Opened in a New Window" (Manually), a numeric field for "Number of Additional Windows a User can have Open Concurrently" (20), a dropdown for "Multiple Window Control Bar Location" (Above Title Bar), and a version number field (1.0.0.57) with checkboxes for "Automatically Increment when Saving" and "Show in Help About Text". The "Last Changed" field contains the value "20080825-133747-VLFPGMLIB". A "Close" button is located at the bottom right.

Web Details	Developer Preferences - LANSA02	Developer Preferences - Web Server 2	User Administration Settings	Instance List Relationships Summary					
Identification	Custom Properties	Visual Styles	Icons	Startup	Commands Enabled	Framework Details	Command Display	Export Design	Help About
Property	Value	Language							
Caption	New version of Your Framework	(ENG)							
Hint:		(ENG)							
Sequence:	1								
Internal Identifier:	VF_SHIPPED								
Unique Identifier:	1								
User Object Name / Type	SHIPPED_FRAMEWORK	Verify Name							
<input checked="" type="checkbox"/>	Show the 'Windows' Menu in this Framework								
<input type="checkbox"/>	Show Current Business Object in Window Title								
Allow this Object to be Opened in a New Window	Manually								
Number of Additional Windows a User can have Open Concurrently:	20								
Multiple Window Control Bar Location:	Above Title Bar								
Your Framework Version Number	1 . 0 . 0 . 57								
<input checked="" type="checkbox"/>	Automatically Increment when Saving								
<input checked="" type="checkbox"/>	Show in Help About Text								
Last Changed	20080825-133747-VLFPGMLIB								

Framework Icon

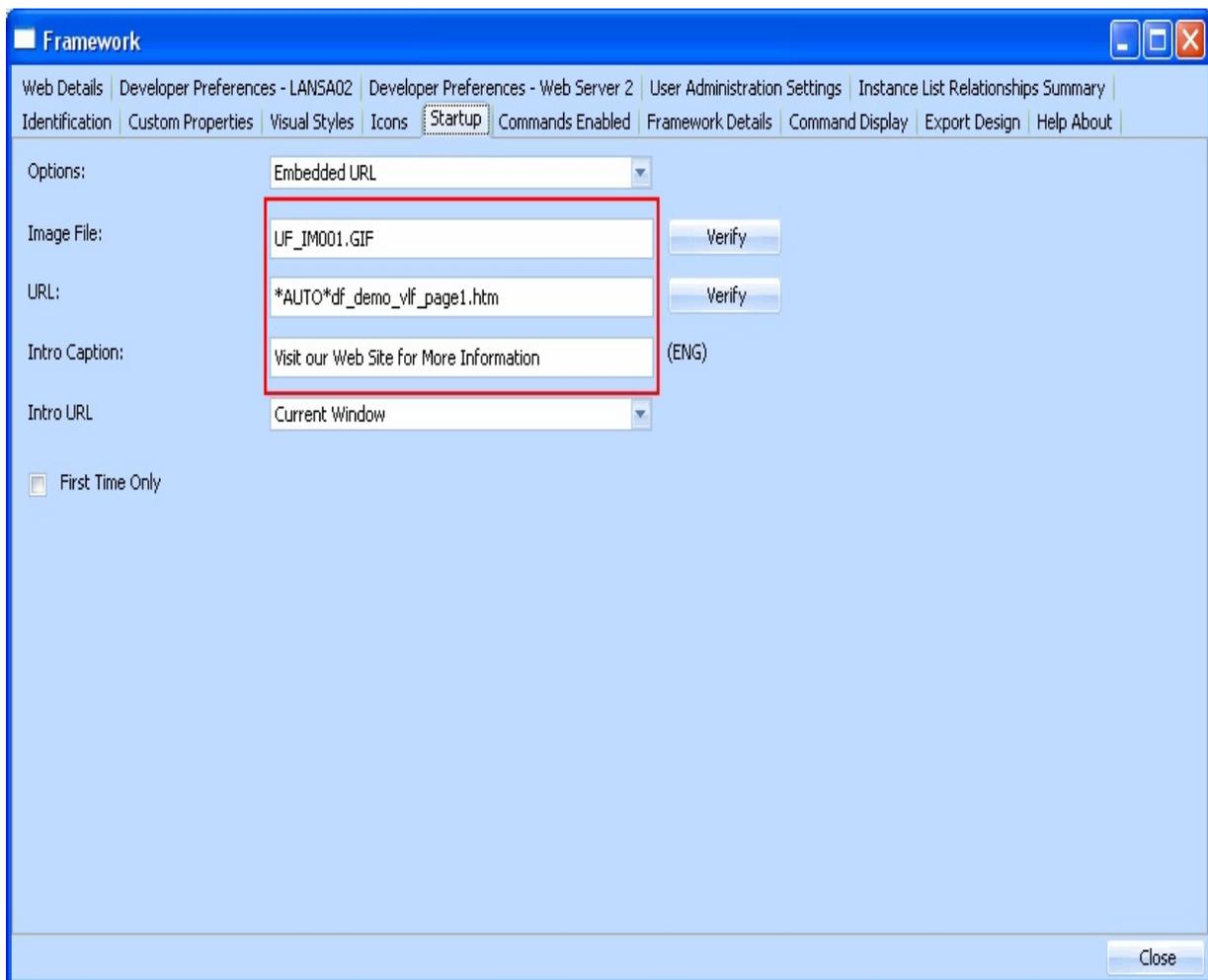
Use the Icons tab to select an icon:



Framework splash image

Change the startup appearance of the application on the Startup tab:

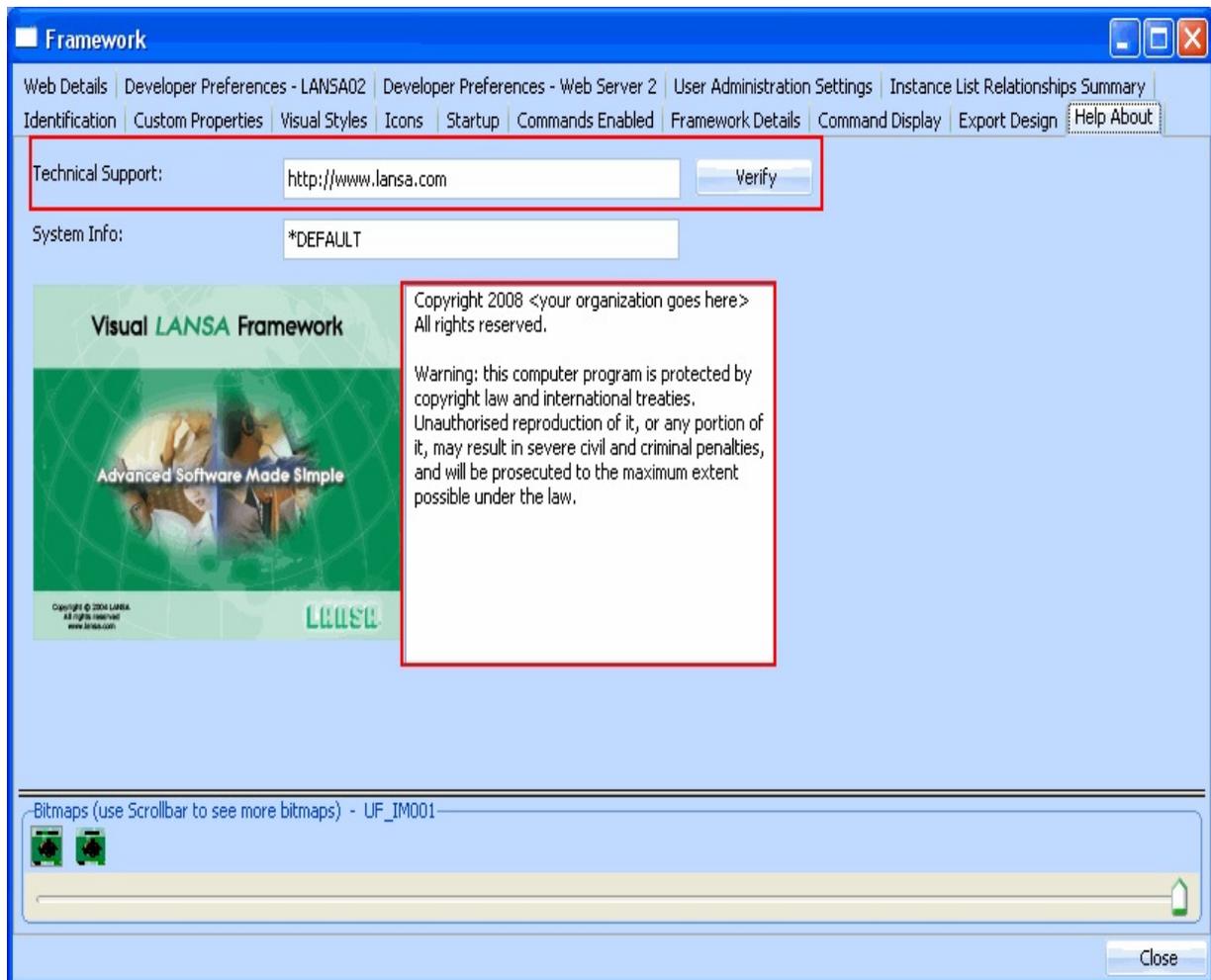
- The splash image is shown after Framework has loaded. You can use any image file in the partition execute directory.
- The URL to link to (below the image)
- The caption to show for the link



The technical support URL

Use the Help About tab to change the help about details:

- The text that will be displayed when the user chooses the Framework menu option About, including Technical Support address and copyright notice
- The image that will be displayed



To use your own bitmap as the about Framework image, you need to create your own LANS A bitmap and load your image into it.

You then need to create your own version of component UF_IB001 and add a line to it to enrol the LANS A bitmap in the Framework. (See the source for reusable part UF_IB001 for instructions on how to do this).

When your bitmap is enrolled in the Framework, you can select it from the list

of bitmaps in the Help About tab.

The Sign on Screen image

You can use any image file in the partition execute directory:

The screenshot shows the 'Framework' application window with the 'User Administration Settings' tab selected. The 'Sign on Form Settings' section is highlighted with a red border. The settings are as follows:

- Authority Settings:**
 - Use Framework Users and Authority
 - Store Users in DBMS Tables VFPPF06/7
 - Store Users in XML File Named:
 - Import Users Imbedded Interface Point:
- Sign on Settings:**
 - End Users must Signon to this Framework:
 - Users Sign on Locally to Use the Framework
 - User Can Change Own Password
 - Users Sign on to a Remote Server to Use the Framework
 - Users May Work Offline if the Remote Server Is Not Available
 - Maximum Signon Attempts Allowed:
 - If Maximum Allowed Sign on Attempts Exceeded:
 - Advise User with a Message
 - Framework Fatal Error
- Sign on Form Settings (highlighted):**
 - Image to Display:
 - Alignment of Image on Form:
 - Height of Image:
 - Width of Image:
- Timeout Settings:**
 - Inactivity Log on timeout (minutes):
 - Inactivity Log off timeout (minutes):

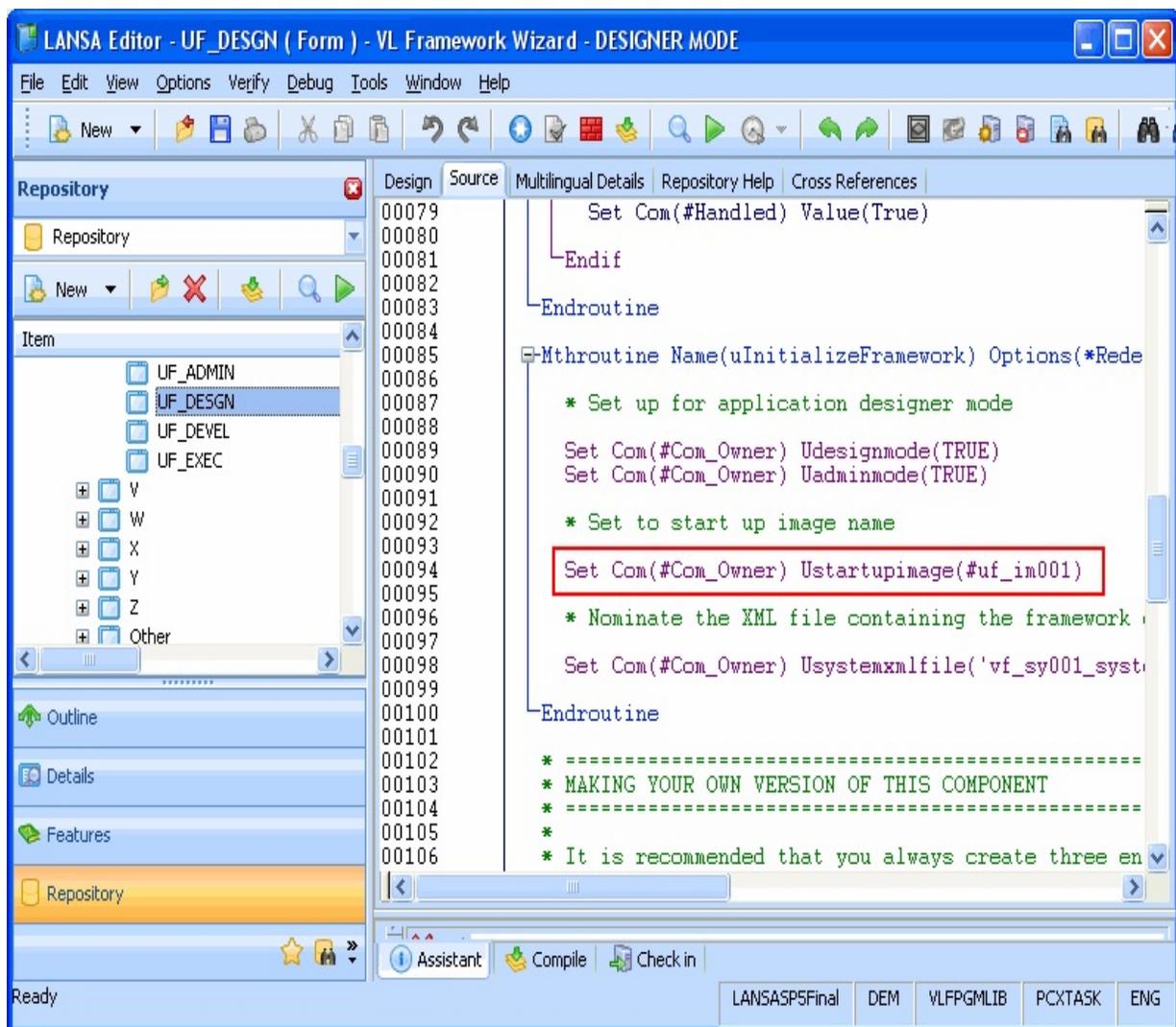
The 'Close' button is located at the bottom right of the dialog box.

First Splash Screen

There is one other change that is not under Framework properties.

This is the image that is first displayed when the Framework starts up. The first splash screen.

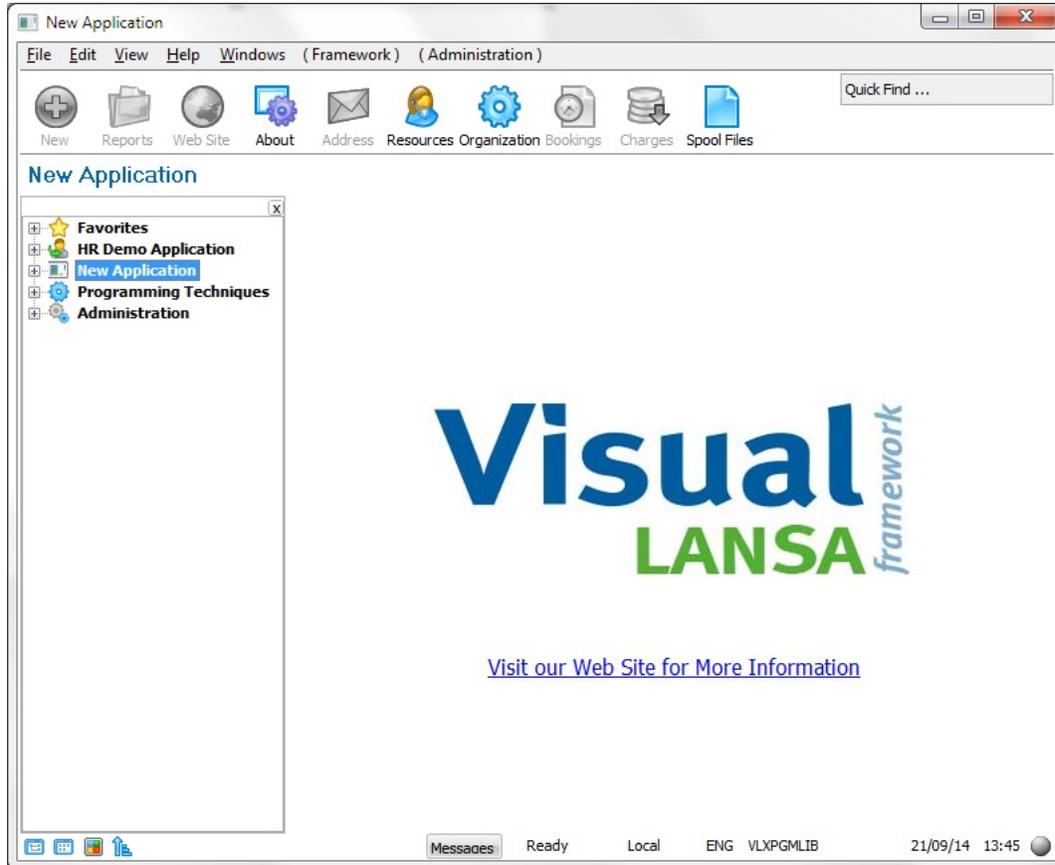
To change this, first create your own LANSAs bitmap and load your image into it. Then modify the entry point forms (UF_DESGN, UF_ADMIN, and UF_EXEC (or their equivalents)), so that the line shown below points to your LANSAs bitmap.



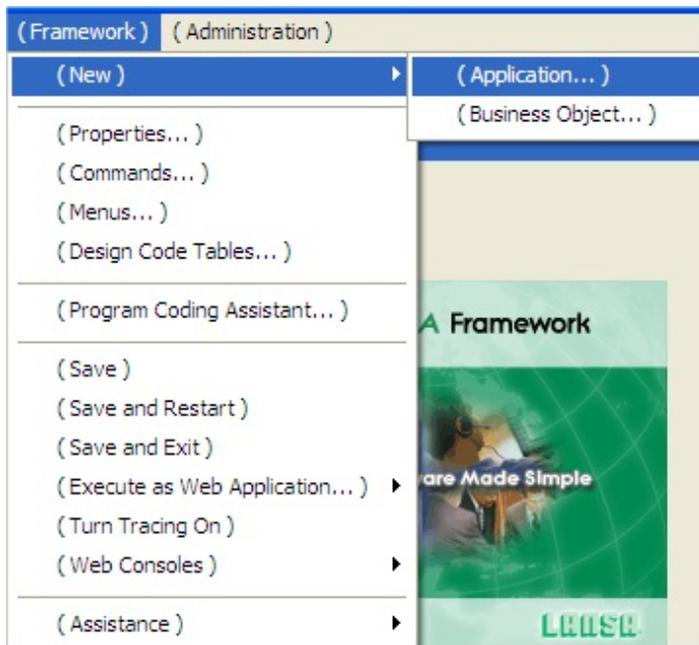
Define Your Application

Tutorial: [VLF001 - Defining Your HR Application](#)

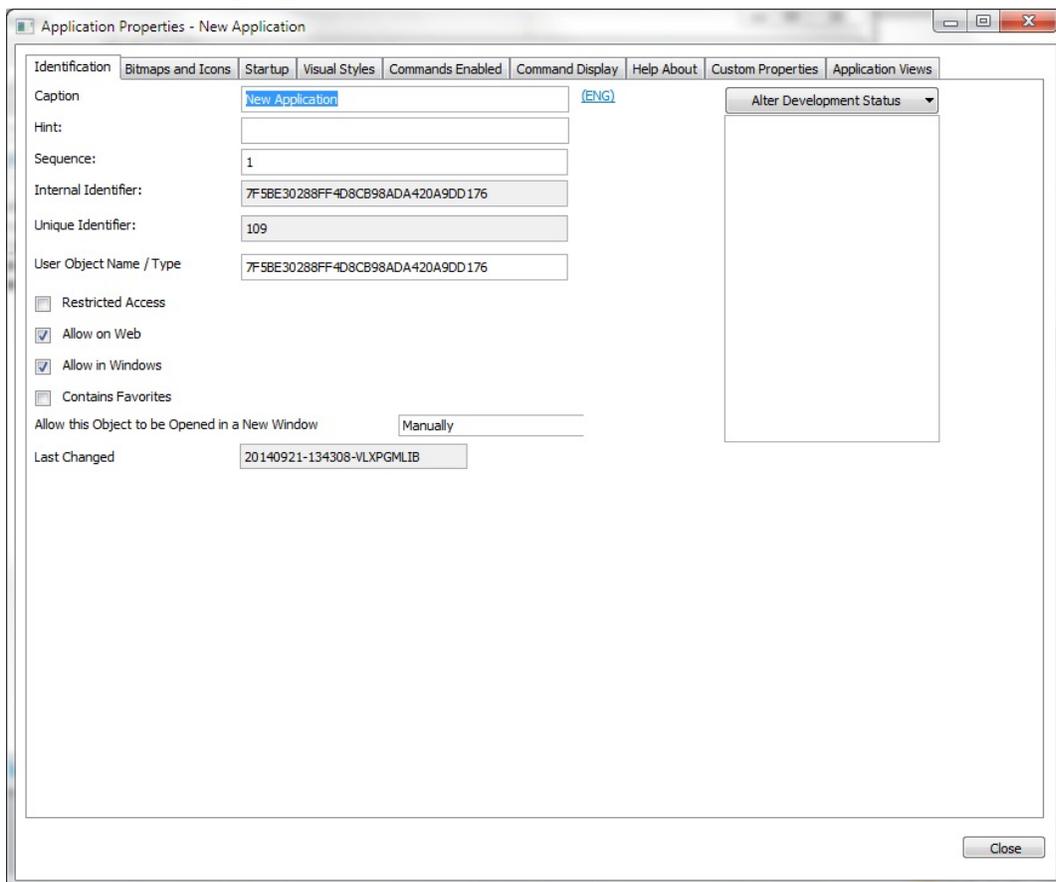
You start by creating the application itself.



To create an application, use the New option in the Framework menu:



And define its properties:

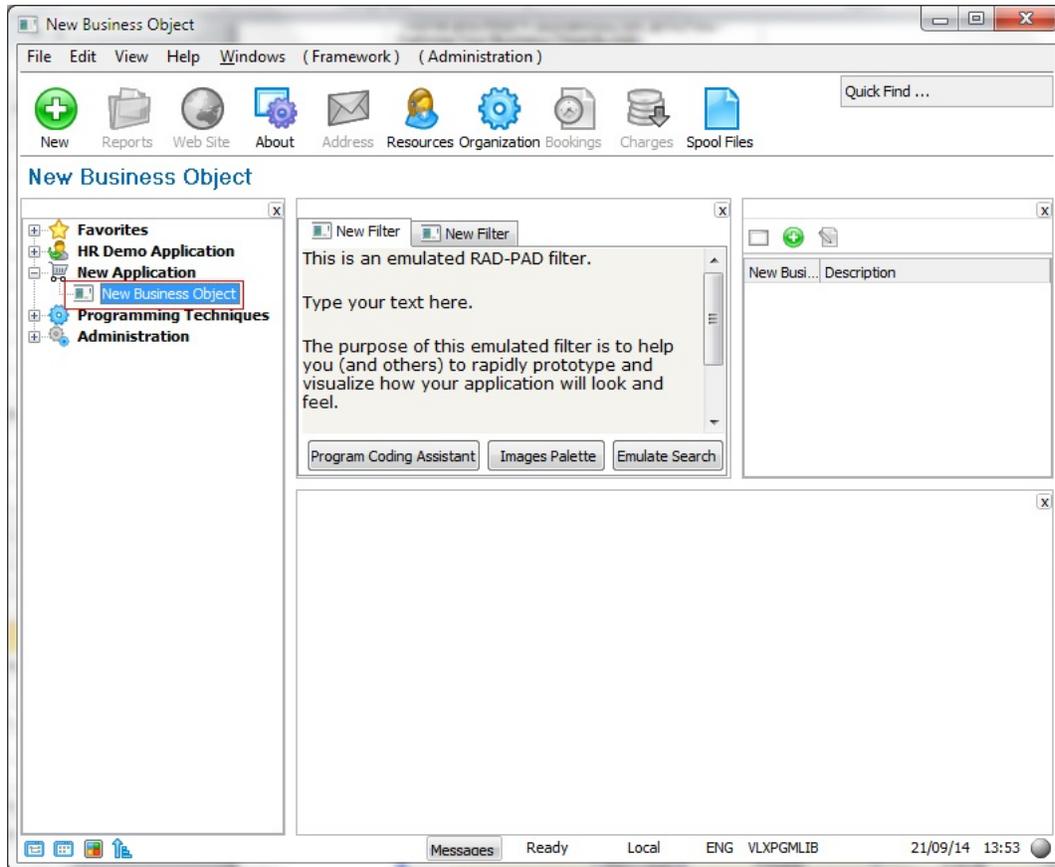


You can change the application properties by selecting the application, right-clicking and choosing the Properties... option from the popup menu.

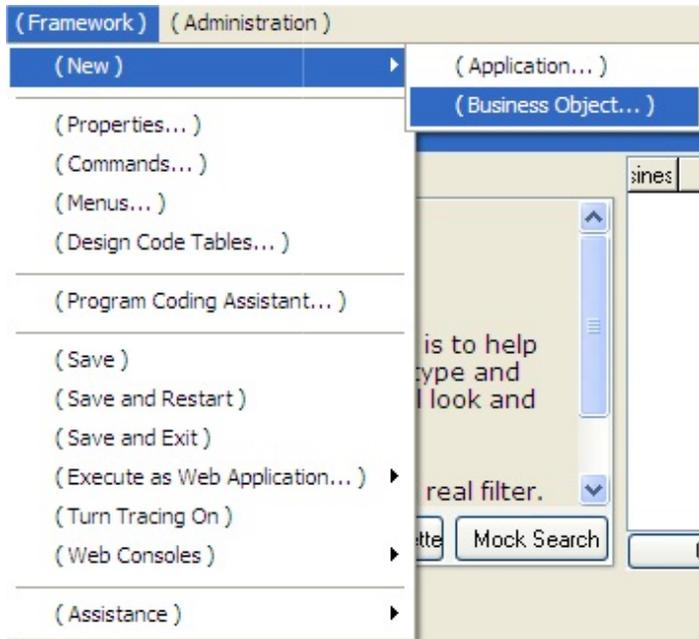
Define Your Business Objects

Tutorial [VLF002 - Defining Your Business Object](#)

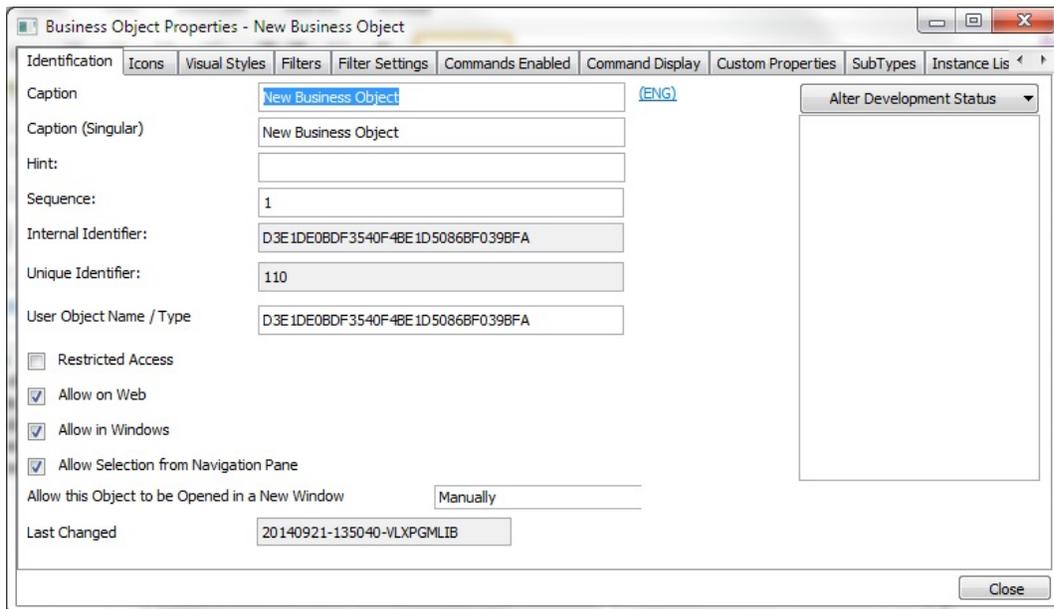
Then you create your business objects.



To create a business object, use the New option in the Framework menu:



And define its properties:

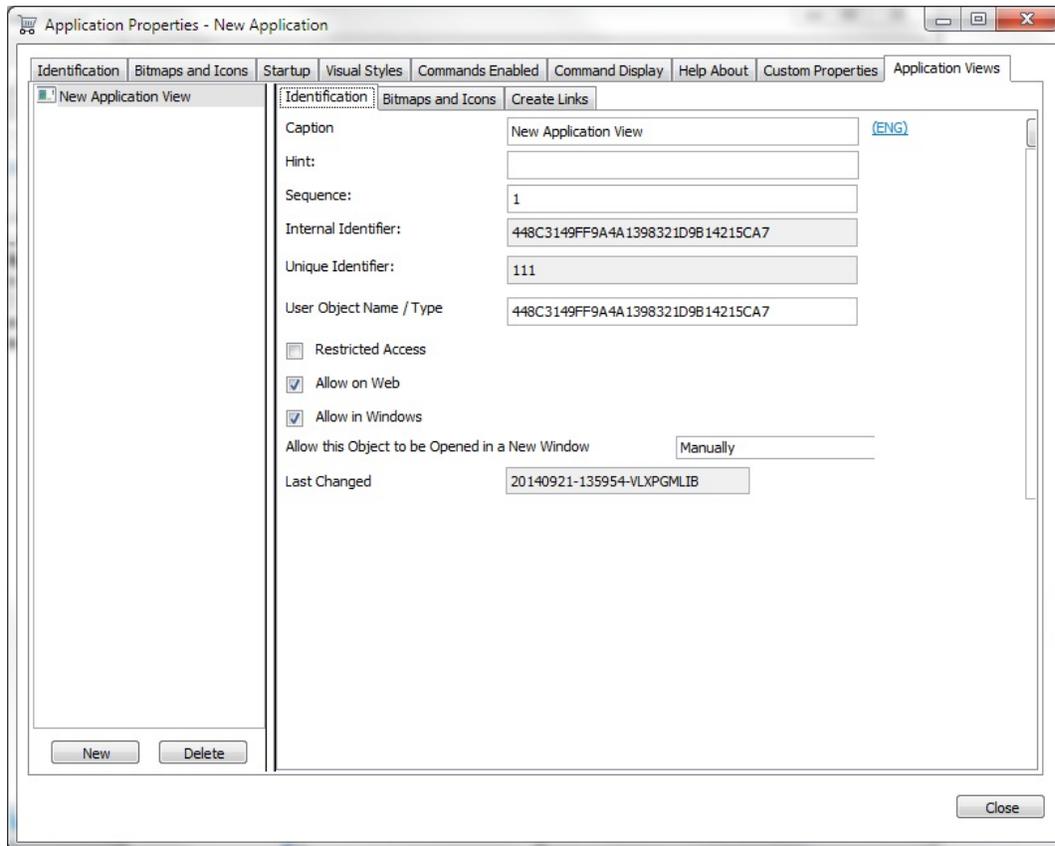


You can change the business object properties by selecting the application, right-clicking and choosing the Properties... option from the popup menu.

Optionally Group Business Objects into Application Views

Now that your Business Objects have been created you can optionally group them as different views of the Application. You can create Application Views at any time during or after an application has been created.

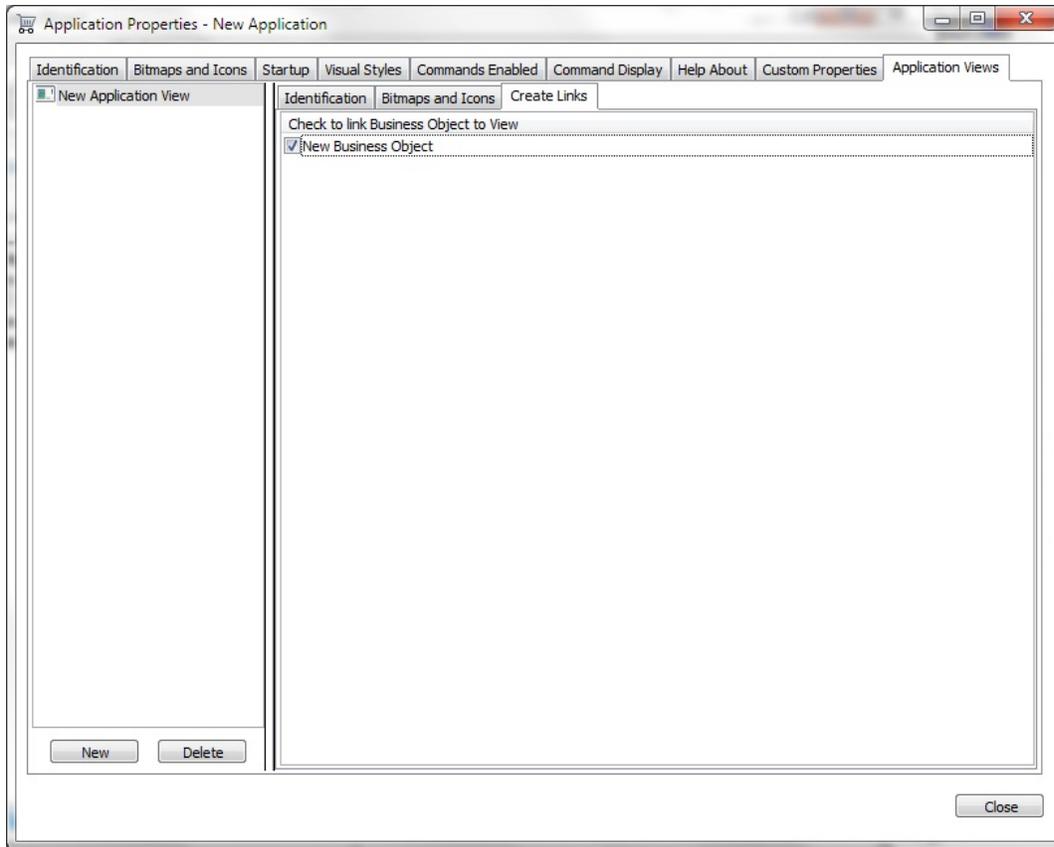
To create an Application View bring up the properties panel for the application and click on the Application Views tab. To create a View, click on the New button:



Specify the view's identification properties.

Using the Bitmaps and Icons tab you can define an icon and a bitmap to display for the view.

Click on the Create Links tab to link Business Objects to the newly created application view. A list of the business objects currently defined for the selected application is displayed:



Check the boxes next to each Business Object you want to include in the currently selected View.

Remarks

- Application views have their own security.

Security in views is independent of the security applied to the business objects defined in them. This means users not allowed to a specific view will not be able to execute the business objects grouped inside that view because the view will not be visible. However, users allowed to a business objects inside a view to which they are not allowed will be able to switch to that business object.

- You can define up to 100 Views for an application. However, more than 10 is considered excessive and will affect the amount of space available to show the views and business objects belonging to them. If more than 10 are required you should restrict the Navigation View Pane to Tree View.

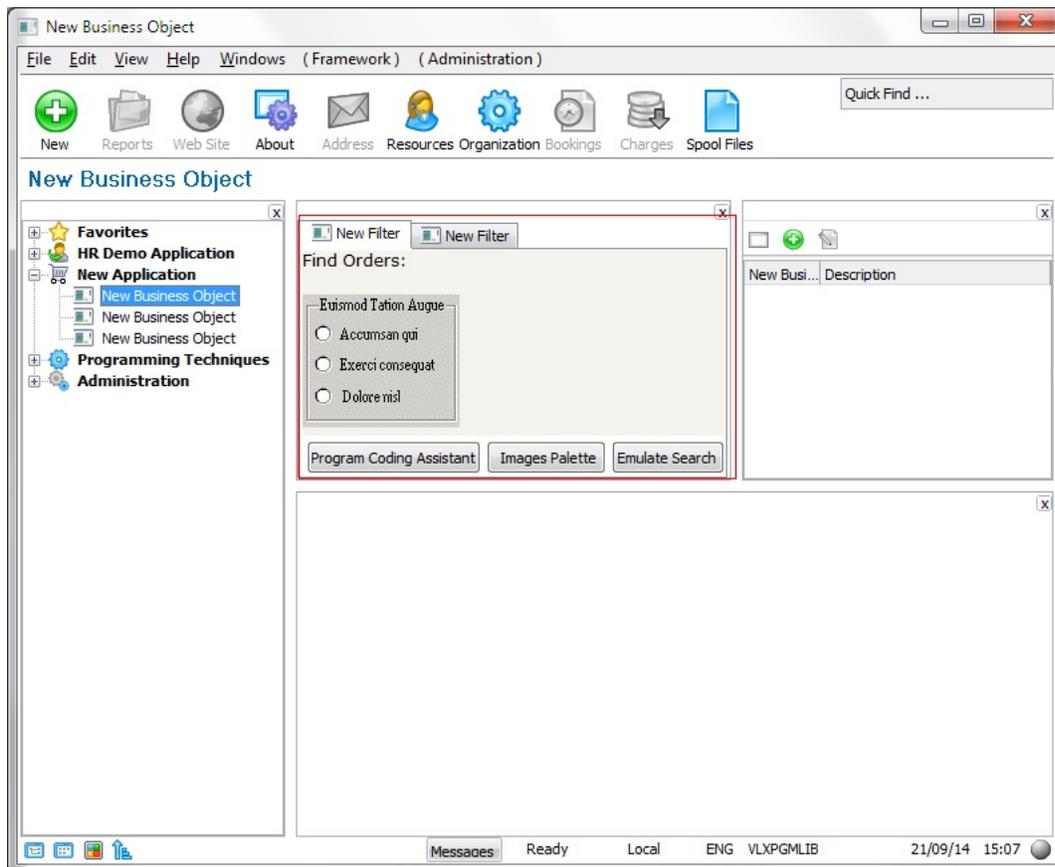
- Once you define a view for an application, all business objects defined for the application must be defined in the same or other views. In this scenario, all business objects not defined in any view will be automatically linked to a new view created when the Framework is saved.

- You can define one business object in more than one view.

Prototype Your Filters

Tutorial [VLF003 - Prototyping Your Filters](#) [Filter](#) [Mock up Filters and Command Handlers](#)

Next you create prototype filters for your business objects. Use the prototype filters or create your own mock-up filters by typing in text or using the Images Palette. You don't yet write any code.



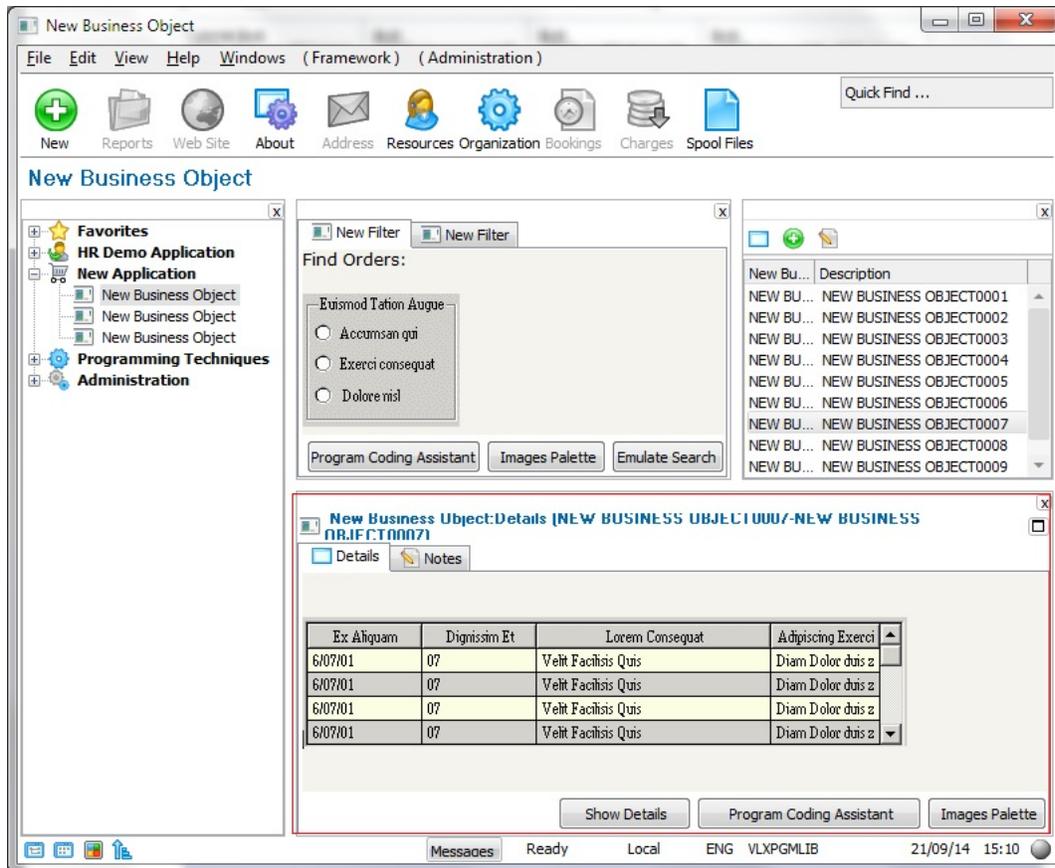
Prototype Your Commands and Their Handlers

Tutorial [VLF004 - Prototyping Your Commands](#)

[Command](#) [Command Handler](#)

[Mock Up Filters and Command Handlers](#)

Then create command handlers for your application and business objects. Use the prototype command handlers or create your own mock-up command handlers. You don't yet write any code.



Some Guidelines for Defining Commands

- Many commands are verbs and some are nouns. Most of the noun commands actually have an implied verb associated with them (e.g.: the command "Picture" is actually "Show me a Picture").
- If commands are actually verbs (real or implied) then you should try to always think about defining commands in the Object → Action context: The user first chooses the Object (i.e.: the Framework, an application or a business object), then they indicate the Action they would like to take on the object by selecting a command.

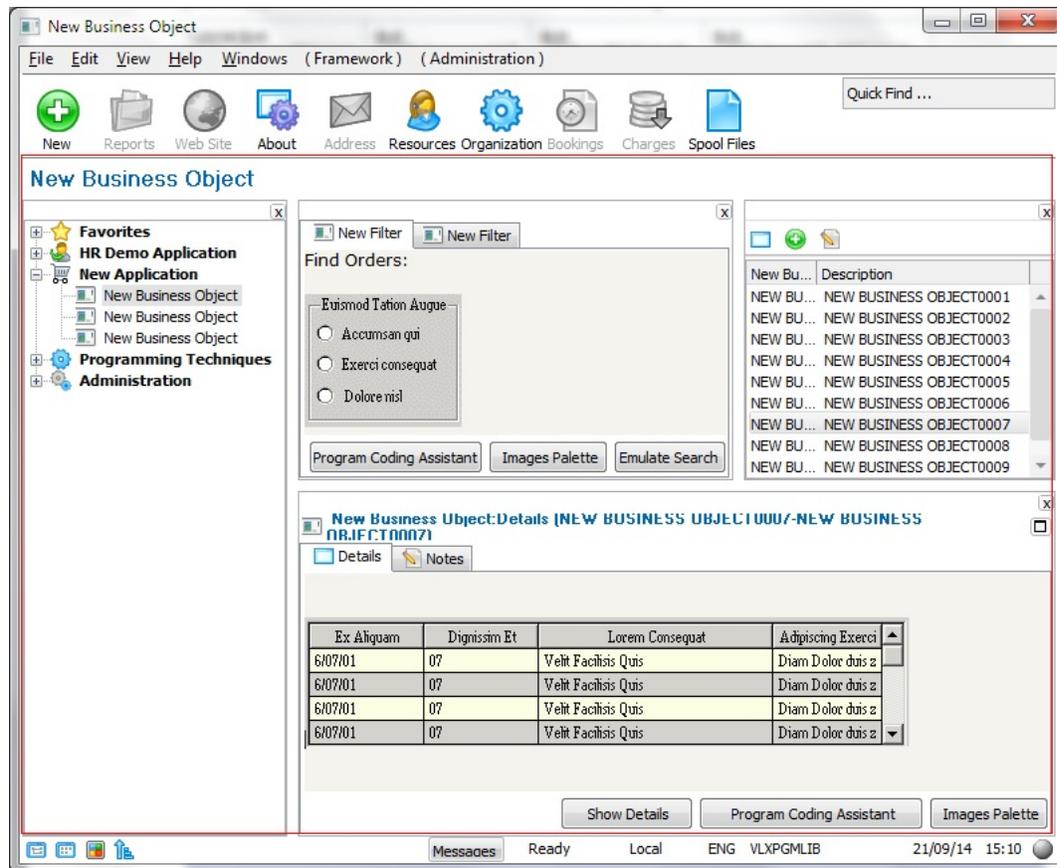
- This is different to the way that many procedural menu based systems work. Classically they are Action → Object orientated.
- You should try to use simple, reusable and generic nouns or verbs for commands.
- For example, use "Details" rather than "Customer Details" as this will allow the command "Details" to be reused with many other objects such as Employees, Products and Orders ... saving you from overcrowding your Framework with "Employee Details", "Product Details" and "Order Details".
- Users will understand the command "Details" applies to the currently selected object (e.g.: Customer, Order, Employee or Product) for the same reason that they understand that "Copy" in MS-PowerPoint means to copy the currently selected object which is why MS-PowerPoint does not have to have "Copy Text", "Copy Picture", "Copy Clip-Art", etc and can have a single "Copy" reusable command.

Validate Your Design

Tutorial [VLF005 - Validating the Prototype](#)

Your prototype is now ready to be tested out and shown to others. It looks and acts like a real application, except that the filters and command handlers do not actually perform any processing.

Visualize and validate your application prototype with users and developers:



Create Your Own Filters

[Framework Programming](#) Tutorials:

[VLF006WIN - Snapping in A Real Windows Filter](#)

[VLF006WAM - Snapping in A Real WAM Web Filter](#)

When you turn your prototype to a real application, you snap in your custom made filters which provide the actual filtering.

WINDOWS: You create the filter as a reusable part, compile it and then snap it in the Framework.

WAM: You create the filter as a Web Application Module, compile it and then snap it in the Framework

Use the [Program Coding Assistant](#) in the Framework to generate the filter code for you or define it manually

Snap it in the Framework:

- Start the Framework executing (as a designer).
- Display the Filters tab for the business object you are working with.
- Select the mock up filter that you wish to replace with your real filter. Click on the Filter Snap-in Settings Tab.
- Use the Windows group box to associate a Windows real filter to the selected Business Object filter. Use the Web Browser group box to associate a WAM real filter to the currently selected Business Object filter.
- Click the radio button of one of the real handler types.
- Type the handler name into the property field if it is known.
- Alternatively, click on the Find button to open the repository search dialogue. Enter a partial name and/or description and click the Find button to restrict the of return repository objects matching those strings.
- Locate your filter handler in the object list and select it by double clicking on it or selecting it and clicking the OK button.
- Check the WAM Component radio button to associate a WAM real filter handler to the currently selected Business Object filter.
- Save the Framework.

Your filter is now snapped into the Framework and usable. Test, modify, debug and recompile your filter as you would any component.

Create Your Own Command Handlers

[Framework Programming](#) Tutorials:

[VLF007WIN - Snapping in A Real Windows Command Handler](#)

[VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

When you turn your prototype to a real application, you snap in your custom made command handlers, which perform the processing in your application.

WINDOWS: You create the command handler as a reusable part, compile it and then snap it in the Framework.

WAM: You create the command handler as a Web Application Module, compile it and then snap it in the Framework

Use the [Program Coding Assistant](#) in the Framework to generate the command handler code for you or define it manually

Snap It in the Framework:

- Start the Framework executing (as a designer).
- Display the property dialog of the object with which the command is associated:

For the Framework	Select menu item Framework and then Properties
For an Application	Double-click on the application.
For a Business Object	Double-click on the business object.

- On the resulting properties dialog click on the Commands Enabled tab.
- Select the command that you wish to replace with your real command handler. Make sure to select whether the command is a Business Object or Instance command, and to set the desired Command Options:
- Use the Windows group box to associate a Windows real filter to the selected Business Object filter. Use the Web Browser group box to associate a WAM real filter to the currently selected Business Object command.
- Click the radio button of one of the real handler types.
- Type the handler name into the property field if it is known.

- Alternatively, click on the Find button to open the repository search dialogue. Enter a partial name and/or description and click the Find button to restrict the of return repository objects matching those strings.
- Locate your command handler in the object list and select it by double clicking on it or selecting it and clicking the OK button.
- Check the WAM Component radio button to associate a WAM real command handler to the currently selected Business Object command.
- Save the Framework.

Your handler is now snapped into the Framework and usable. Test, modify, debug and recompile your filter as you would any component.

Optionally Create Your Own Instance List

[Adding Additional Columns to Instance Lists](#)

[Tutorial: VLF009WIN - Adding Instance List Columns in Windows Applications](#)

[Tutorial: VLF009WEB - Adding Instance List Columns in Web Applications](#)

The Framework provides a standard [Instance List](#) that displays your business object instances. If the shipped instance list browser will not do exactly what you need, you can write your own. A Code Assistant is shipped with the Framework that will generate a basic instance list browser for you.

The Code Assistant creates code for flat visualization controls such as list views and grids. If you want to present a tree, see the following section for an example of a snap in instance list browser displaying a tree.

If you create your own snap in instance list browser reusable part, you need to specify its name to the Framework on the Instance List / Relations tab here:

Enable Clear List Button

Double click for default command

Save and Restore Instance Lists

Snap in Instance List Browser

XXXXXXXXXX

Deploy the Application

Tutorial: [Tutorials for Deployment](#)

Lastly you need to deploy your application for your end-users.

How you go about this varies according to what the application contains.

Your application may contain components in one or more of these broad categories:

Windows client components

These are objects that need to be installed onto your user' desktops (or on a server to which they have high speed access) for your application to execute. For example all your Windows filters and command handlers would fall into this category. Refer to the [Tutorials for Deployment](#).

Windows or System i server based components that support your Windows clients

These are objects that need to be installed onto your server system(s) to support your Windows clients. For example your database and any remote procedures you have created would fall into this category.

System i based Web Browser applications

These are mainly Process and Function objects in executable form to import into the partition the Visual LANSA Framework will be deployed to. They are also image files, Java Script files, and other types of files that are typically deployed into the System i IFS locations that you specified when configuring the Developer Preferences for the associated web server.

Windows Web Browser applications

These are mainly Process and Function objects in executable form to import into the partition the Visual LANSA Framework will be deployed to. They are also image files, Java Script files, and other type of files that are typically deployed into the Windows web server locations that you specified when configuring the Developer Preferences for the associated web server.

Essentially the deployment steps you need to perform are no different to deploying any Visual LANSA or LANSA for the Web application. You just need to include some additional Visual LANSA Framework components that will support the execution of your application in the deployed environment. Refer to [What is Included in the Framework?](#) for details of the Visual LANSA Framework objects that you may need to include.

Tutorials

The Visual LANSAs Framework Tutorials are a set of exercises designed to introduce and reinforce the fundamental skills required to build applications with the Framework.

The tutorials guide you in creating a sample Human Resources application.

Prototype

[VLF000 - Execute Framework Application](#)

[VLF001 - Defining Your HR Application](#)

[VLF002 - Defining Your Business Objects](#)

[VLF003 - Prototyping Your Filters](#)

[VLF004 - Prototyping Your Commands](#)

[VLF005 - Validating the Prototype](#)

Windows Applications

[VLF006WIN - Snapping in A Real Windows Filter](#)

[VLF007WIN - Snapping in A Real Windows Command Handler](#)

[VLF009WIN - Adding Instance List Columns in Windows Applications](#)

[VLF010WIN - Creating a Mini Filter](#)

[VLF011WIN - Creating a Parent Child Instance List](#)

[VLF012WIN - Controlling Navigation Using Switching and the Virtual Clipboard](#)

[VLF013WIN - Signaling Events](#)

[VLF014WIN - Debugging/Tracing](#)

WAM Web Browser Applications

[VLF006WAM - Snapping in A Real WAM Web Filter](#)

[VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

[VLF009WAM - Adding Instance List Columns in WAM Applications](#)

[VLF011WAM - Creating a Parent Child Instance List](#)

[VLF012WAM - Controlling Navigation Using Switching and the Virtual Clipboard](#)

[VLF013WAM - Signaling Events](#)

[VLF014WAM - Debugging/Tracing](#)

Deployment

[VLF008WIN - Deploying the Windows Framework](#)

See [Before You Use the Tutorials](#)

Please send your comments and suggestions to LANSAs Support at:
lansasupport@lansa.com.au.

Disclaimer: While every effort has been made to ensure that the information in this material is accurate, in no event shall LANSAs be liable for any damages arising from its use. LANSAs makes no warranties, expressed or implied.

Before You Use the Tutorials

Who Should Use the Tutorials?

Tutorials can be used by novice or experienced LANSAs developers who wish to learn how to use the Visual LANSAs Framework. Developers should have completed the Visual LANSAs training course or the equivalent. No advanced Visual LANSAs knowledge is required. LANSAs for the Web training is required if you are using the Framework for Web development.

How Do I Use the Tutorials?

It is recommended that you complete the Tutorials in sequence. Complete the exercises related to the style of application that you are creating. If you are only creating Windows applications, you may wish to skip the WAM related exercises.

To allow for more than one developer to use the tutorials, all LANSAs object names will be prefixed with **iii**. You may use any three characters, such as the initials of your name, for the **iii** characters. For example, if your name is John David Smith you can use the characters **JDS**. When asked to create a component named **iiiCOM01**, you will create a component named **JDSCOM01**. Always remember to replace **iii** with your unique 3 characters.

If you are using an unlicensed or trial version of Visual LANSAs, you must use **DEM** to replace *iii*. When asked to create a component named **iiiCOM01**, you will create a component named **DEMCOM01**.

What Partition Should I Use?

It is recommended that you use the DEM partition for the tutorial. The DEM system contains the Personnel System demonstration and all required files used by the tutorial.

Tutorial Installation

In order to use the Tutorials, you must have the Visual LANSAs Framework installed in the partition (installed by Partition Initialization).

The tutorials require the Personnel Demonstration System files (installed by Partition Initialization).

How Many Developers Can Use the Training?

There is no limit on the number of developers who may use the training at the same time. However, it is important that each developer has a unique **iii** identifier for their work.

Your Feedback

Your feedback regarding these tutorials will help us improve the overall quality of the LANSAs documentation and training. Please e-mail your comments to lansatraining@LANSA.com.au

Tutorials for Prototyping

Applies to **Windows** and **WAM** applications.

Includes:

[VLF000 - Execute Framework Application](#)

[VLF001 - Defining Your HR Application](#)

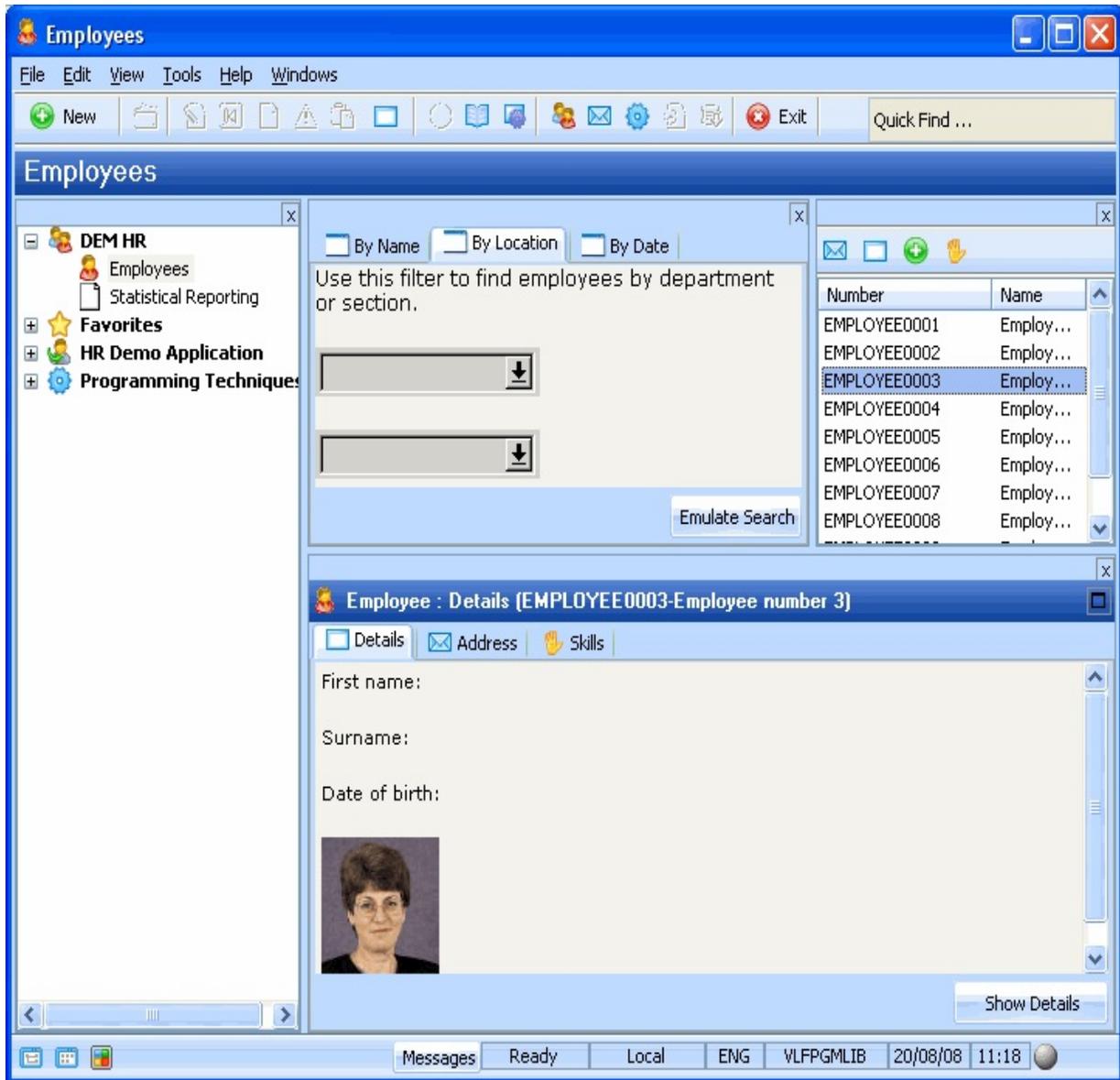
[VLF002 - Defining Your Business Objects](#)

[VLF003 - Prototyping Your Filters](#)

[VLF004 - Prototyping Your Commands](#)

[VLF005 - Validating the Prototype](#)

You will step through the prototyping of a small HR (Human Resources) application. The finished application prototype will appear something like this:



This prototype will contain:

- An **application** called HR which contains two **Business Objects**: Employees and Statistical Reporting.
- Employees business object has three **Filters** (By Name, By Location and By Date) and command handlers for employee Details, Address and Skills. It also has a New command which allows the end-user to define a new employee.
- Statistical Reporting business object contains two reporting commands: Weekly Reports and Monthly Reports.

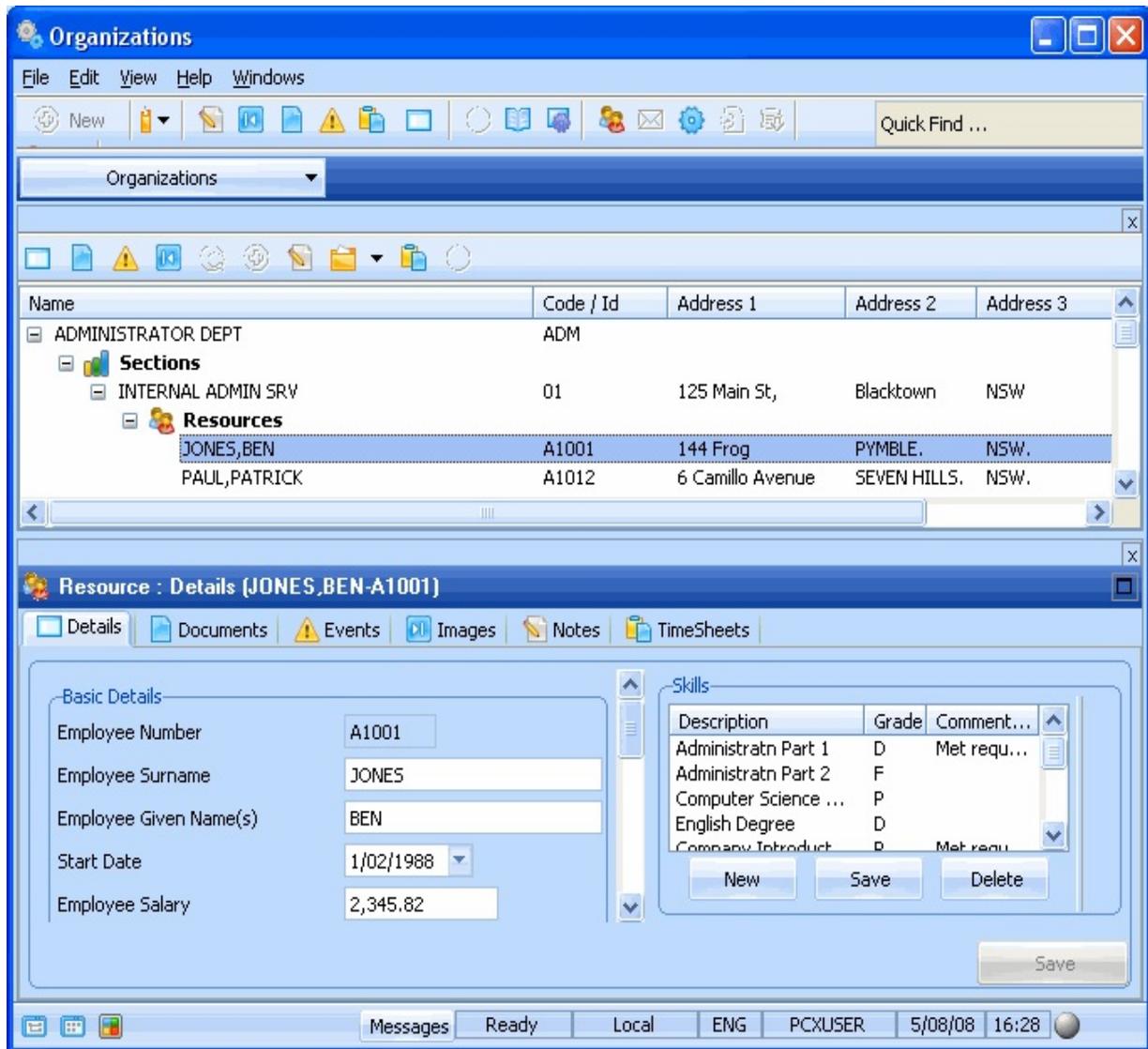
Prerequisites

Make sure the RAD-PAD file format is set to HTML. To check:

- Choose the Properties... option of the Framework menu. The Framework properties dialog is displayed.
- Bring up the Framework Details tab.

VLF000 - Execute Framework Application Objectives

- To execute a finished application in the Framework.
- To become familiar with the look and feel of Framework-based applications.
- To introduce some key concepts used by the Visual LANSA Framework when building applications.



To achieve this objective, you will complete the following steps:

- Step 1. Execute the Visual LANSA Framework
- Step 2. Execute an Application
- Step 3. Using Filters to Find an Employee

- [Step 4. Using Commands and Command Handlers](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

- Check that you have met the prerequisites for the Visual LANSA Framework.

Step 1. Execute the Visual LANSAs Framework

1. Start Visual LANSAs.

Log on to the DEM partition as follows:

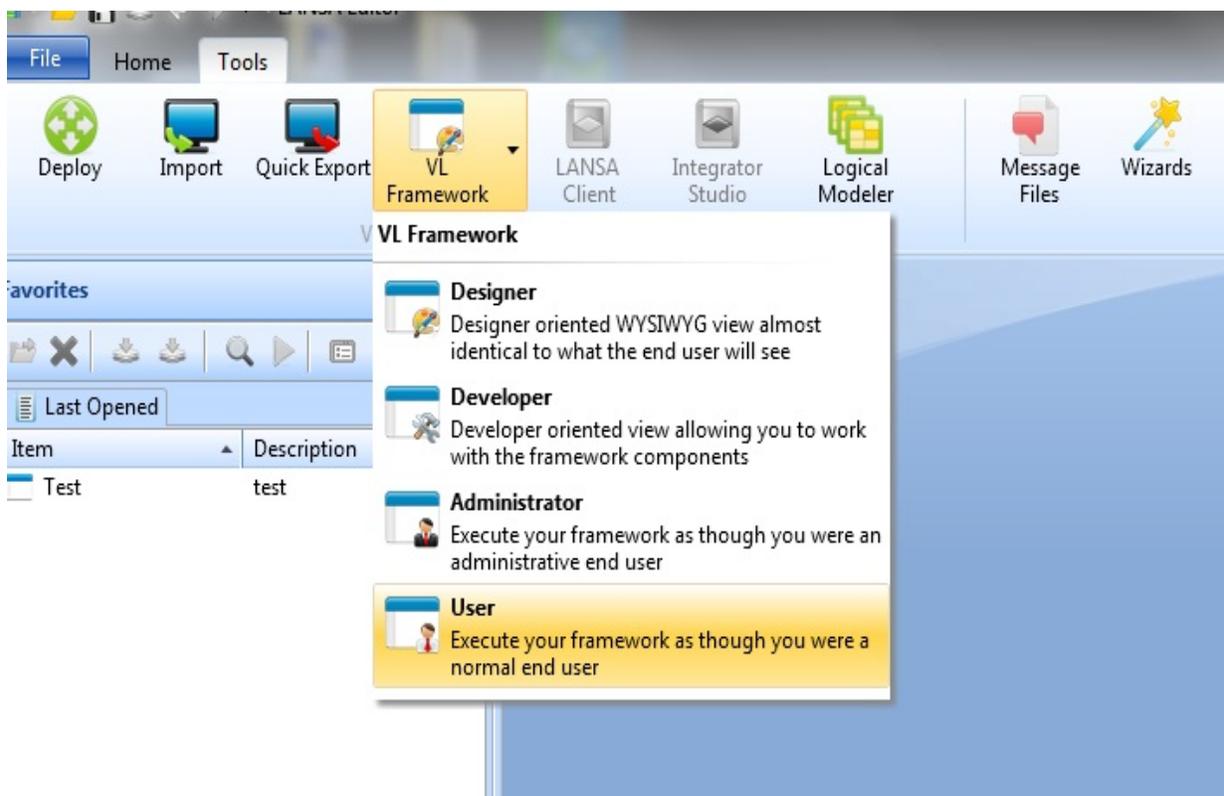
User ID: **PCXUSER**

Password: **PCXUSER**

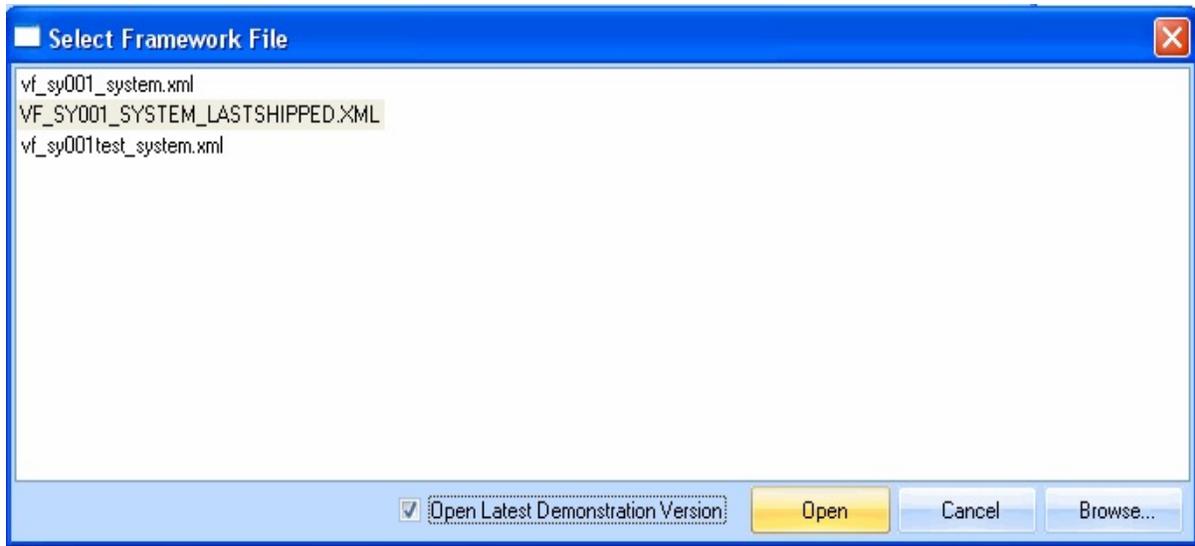
Task ID: **PCXTASK**

Partition: **DEM**

2. Using the Tools tab in the ribbon, select the VL Framework as User option.



3. In the Select Framework File dialog select the option Open Latest Demonstration Version.



The Framework uses XML files to store the definition of your systems. The file `vf_sy001_system_lastshipped.xml` always contains the latest demonstration system.

Note that if you only have one Framework file, this dialog is not displayed. The Framework window will appear.



Step 2. Execute an Application

In this step, you will execute a shipped sample application. You will be introduced to [Business Objects](#), [Filters](#), [Instance Lists](#), [Commands](#) and [Command Handlers](#).

1. The applications in the Framework are shown on the left.

As you expand the different applications, you can see the business objects associated with them.

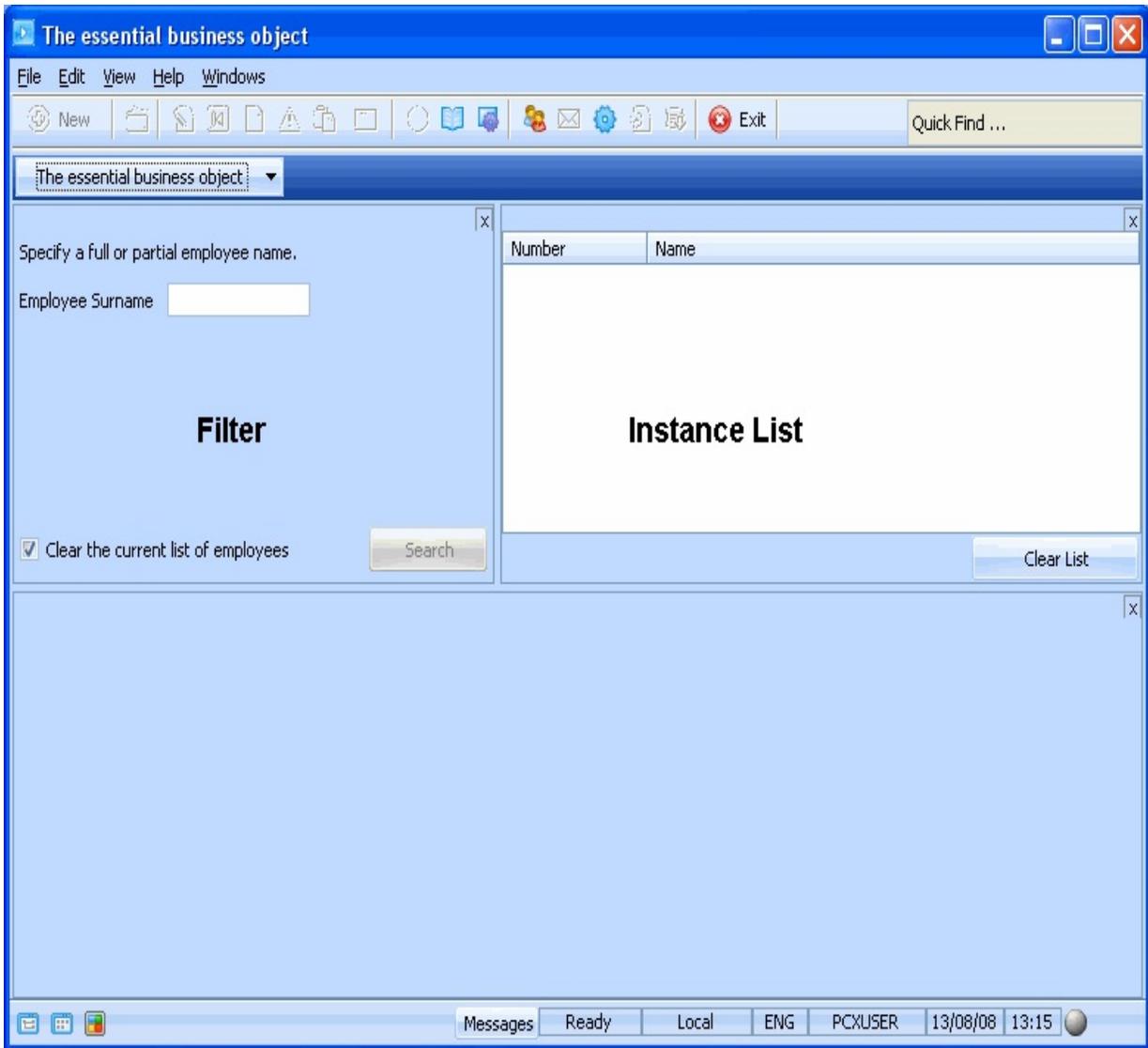
2. Select the Programming Techniques application.
3. Select The Essential business object.

Two new panels will appear.

Use the buttons on the bottom left to show the navigation tree as a button.

The left panel is the filter which is used to search through the employee data.

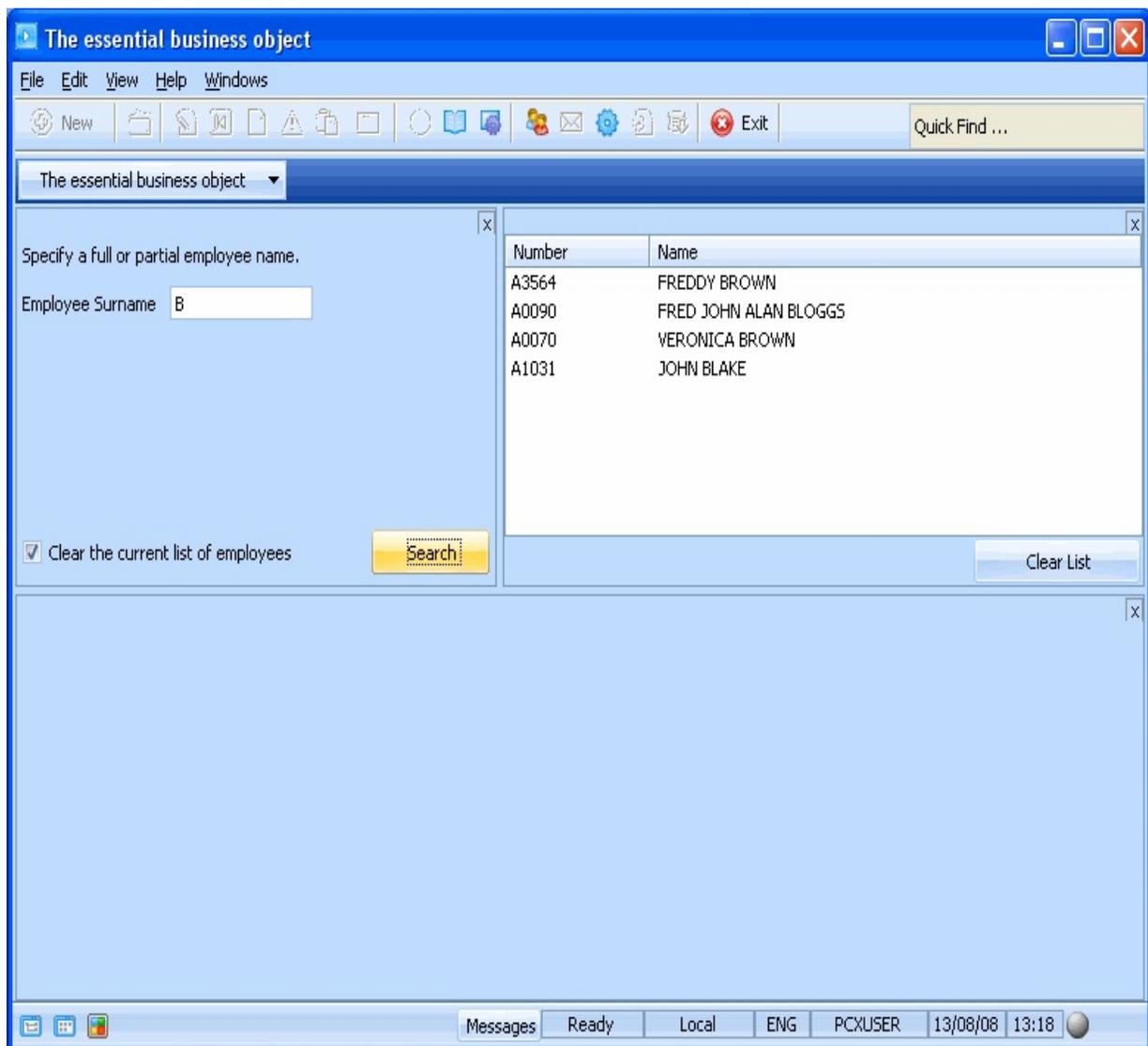
The right panel will show an instance list with the results of an employee search.



Step 3. Using Filters to Find an Employee

In this step, you will use a [Filter](#) to find employees. Filters allow you to search and sort the items in a business object. After an end-user has selected the employee business object, they typically want to locate a specific employee or list of employees.

1. Enter the letter B in the Employee Surname field and click the Search button. The instance list displays all employees whose surname starts with B.

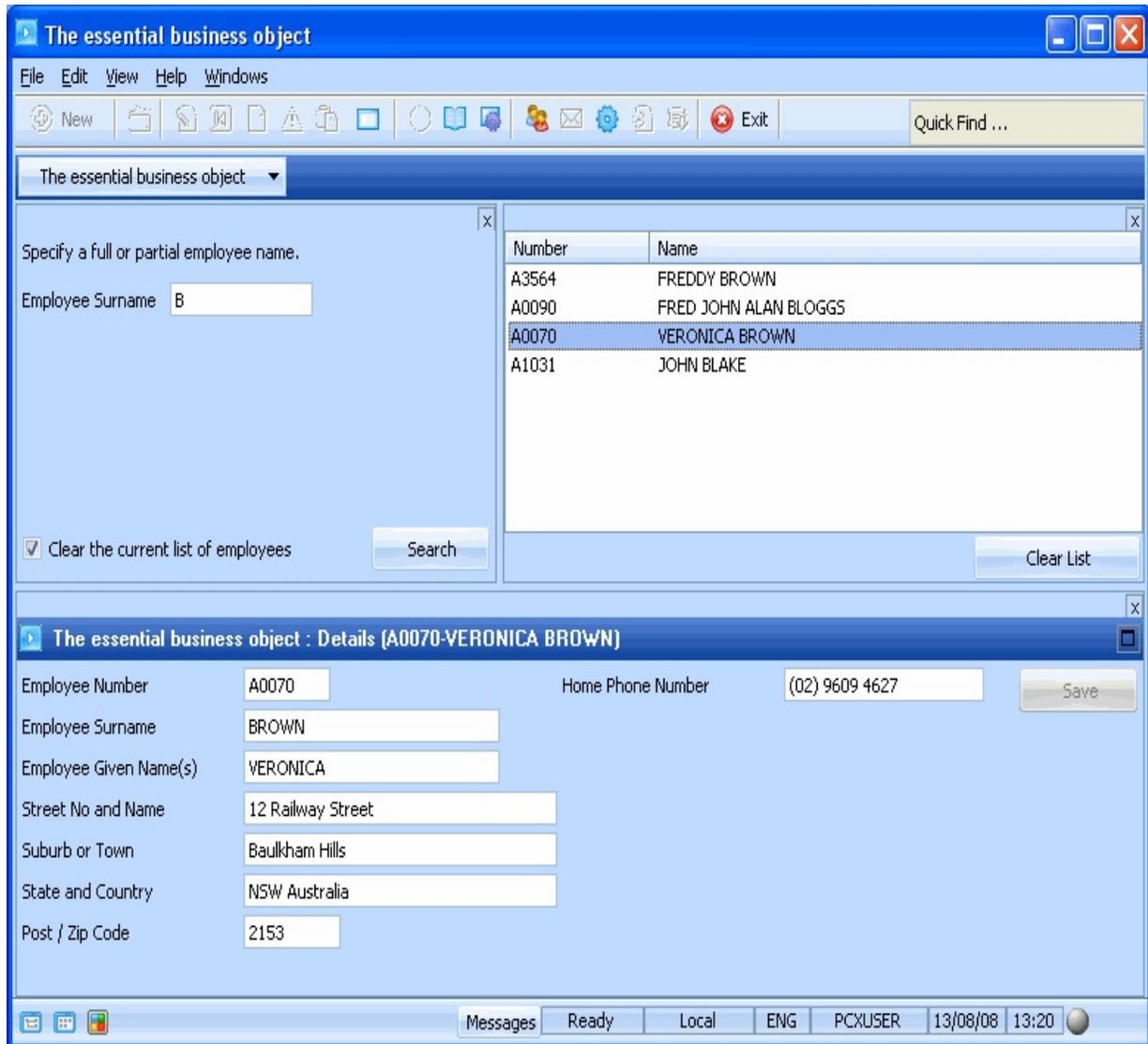


Step 4. Using Commands and Command Handlers

In this step, you will select an employee and review the [Commands](#) or actions which can be performed for the employee.

1. In the instance list, select the employee Veronica Brown.

When an employee has been selected, the Basic details of the employee will appear in the bottom panel.



By default, the Details command has been executed. The Details command handler displays the employee details.

2. Select the File menu and choose the Exit option to close the Visual LANSA Framework application.

Summary

Important Observations

- The Visual LANSA Framework can be executed as a Visual LANSA form. Refer to [VLF005 - Validating the Prototype](#).
- The Visual LANSA Framework provides a consistent application interface. It is very easy to use, flexible and can be customized by the end-user.

Tips & Techniques

- The end-user has the ability to fully customize the appearance of the application within the Framework. For example, the end-user can position the panels in the Framework or can float the panels as separate windows. These capabilities are part of the Framework and are not coded by the developer.
- The Visual LANSA Framework allows the end-user to perform actions in many different ways. Commands can be executed using menus, toolbar icons and pop-up menus.

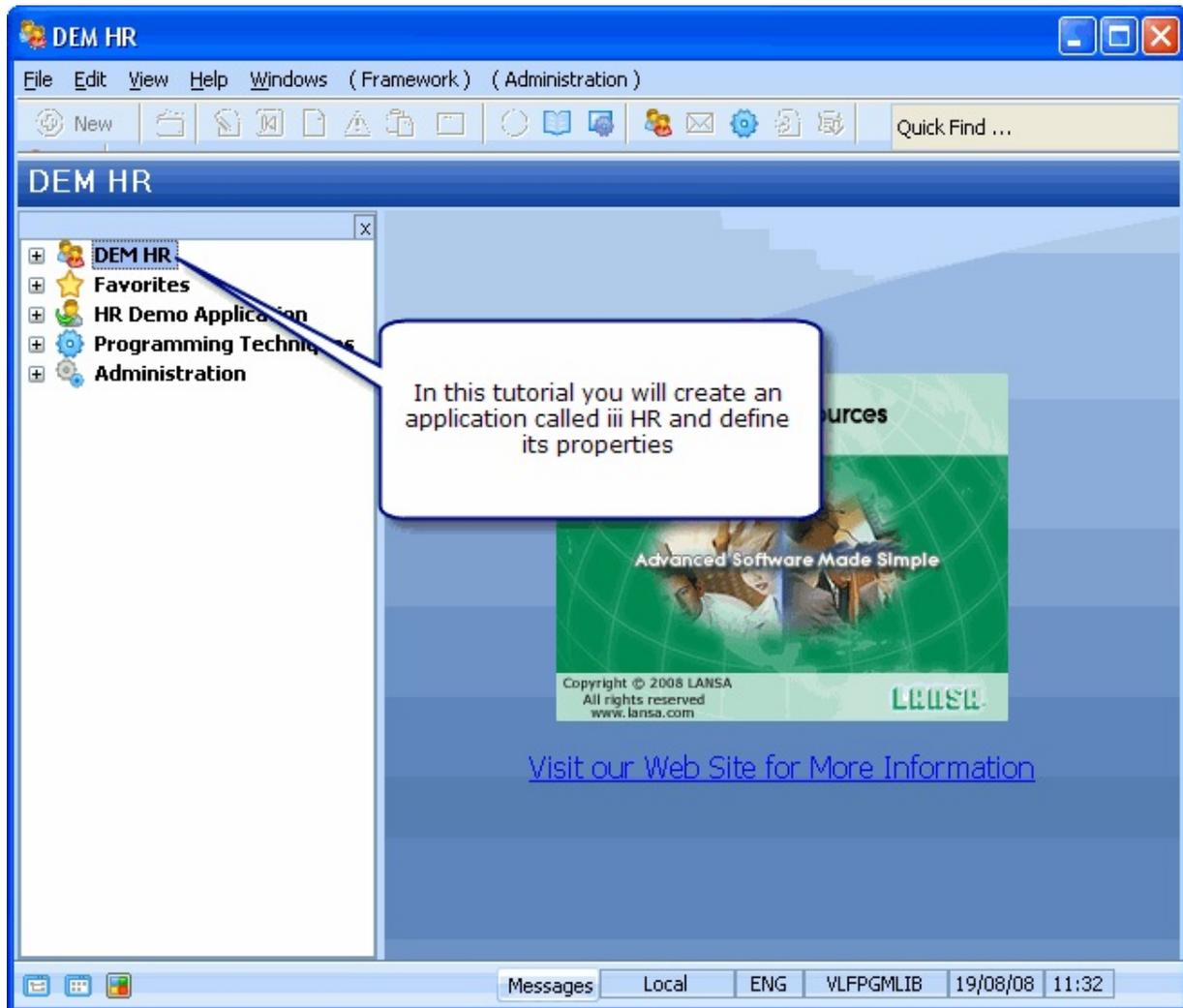
What I Should Know

- How to execute the Visual LANSA Framework as an end-user.
- How to execute an application created in the Visual LANSA Framework.
- What are some of the features supported by the Framework.
- What are applications, business objects, filters, instance lists, commands and command handlers.

VLF001 - Defining Your HR Application

Objective

- To learn how to define an application in the Framework.
- To identify some of the properties of an application.



To achieve this objective, you will complete the following steps:

- Step 1. Create a New Application
- Step 2. Specify Identification Options
- Step 3. Specify Startup Options
- Step 4. View Commands Enabled
- Step 5. Execute the About Command
- Step 6. View Overall Themes

- [Summary](#)

Before You Begin

You may wish to review:

- [Application](#) in [Key Concepts](#).

In order to complete this tutorial, you must have completed the following:

- Check that you have met the prerequisites for the [Tutorials](#).

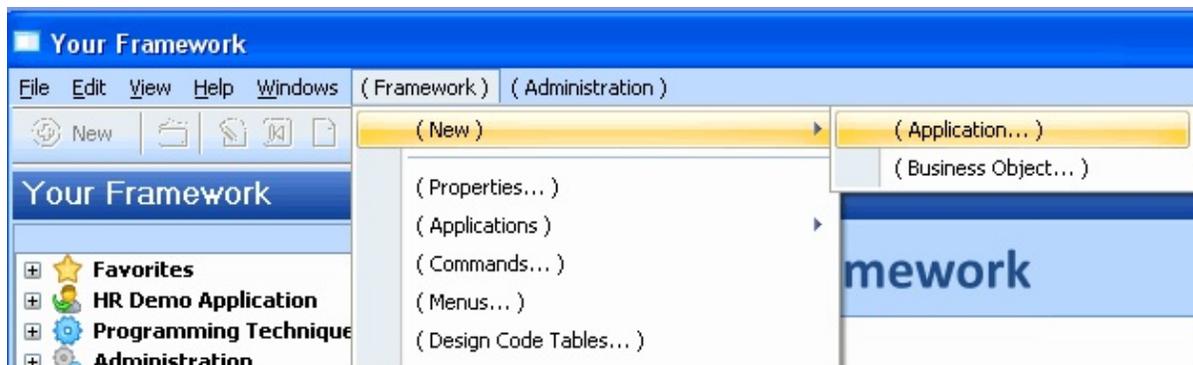
Step 1. Create a New Application

In this step, you will start the Visual LANSAs Framework as a designer and create a new application in the Visual LANSAs Framework.

1. If you have not already done so, start Visual LANSAs and log on to the DEM partition.
2. Using the Tools tab in the ribbon, select the VL Framework and then Designer option.

The Help Assistant and Tutorial Dialog may appear depending on your settings. (Notice: When the Framework is executed as an end-user, this dialog is not displayed.)

3. If it does, do not tick Start the Help Assistant or the Start the Tutorial check boxes (when you want context-sensitive help use F1). Click OK. The Framework window appears.
4. Maximize the Framework window.
5. Select the (Framework) menu and choose New, Application. You may want to have a look at [Menu Options in Brackets](#) .

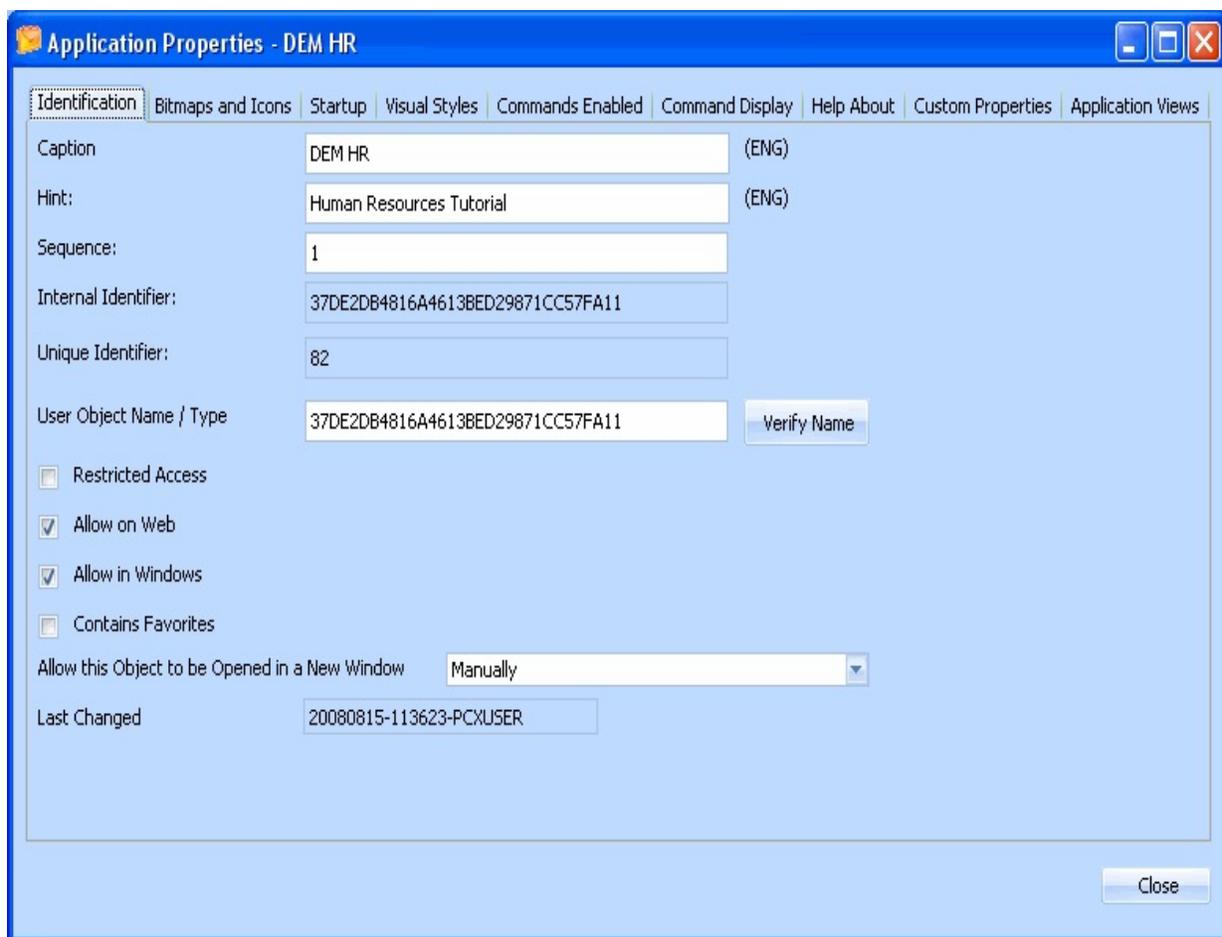


The Application Properties dialog box of the new application is displayed.

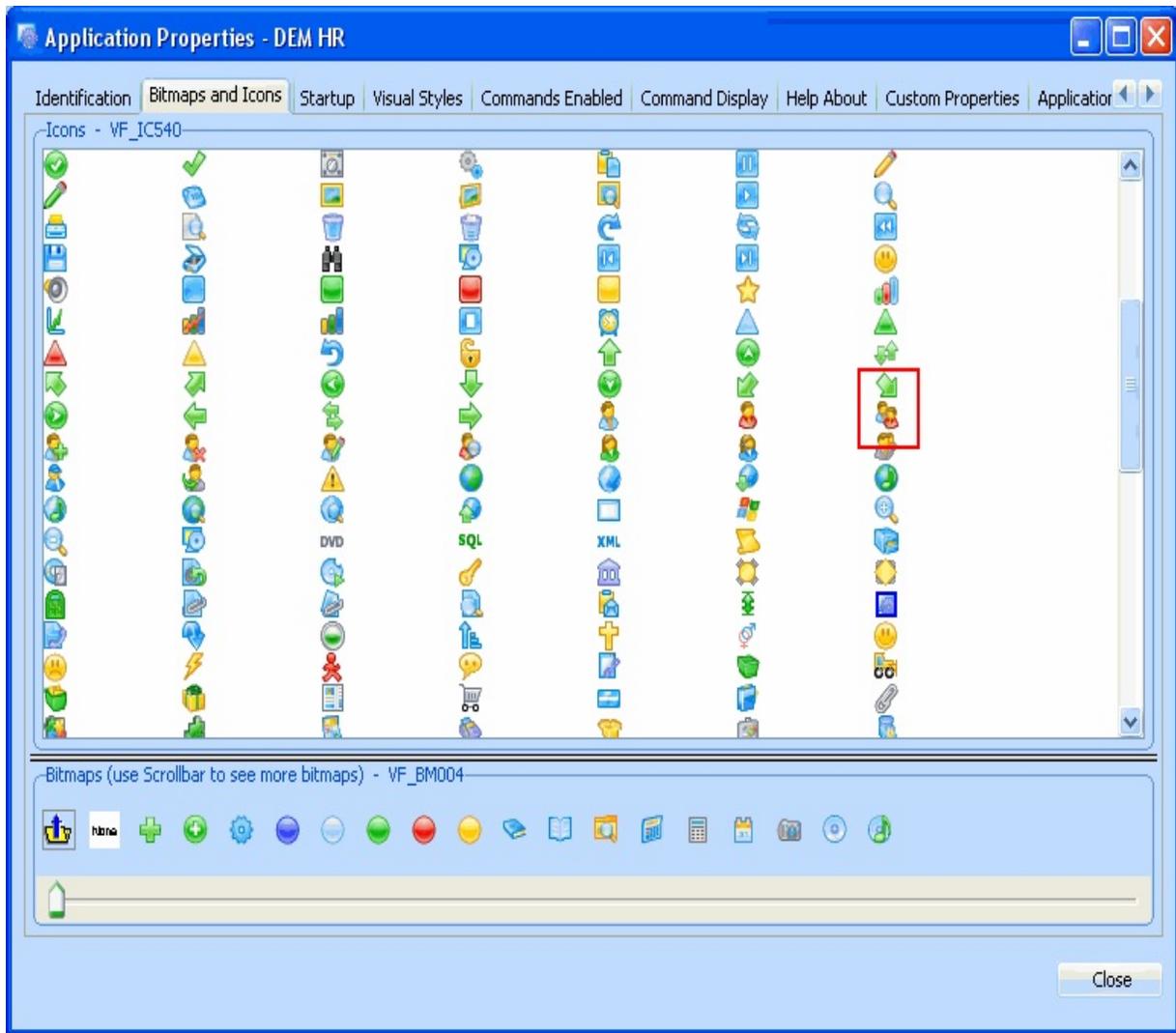
Step 2. Specify Identification Options

In this step, you will identify the application in the Framework by assigning properties such as captions and bitmaps.

1. Select the Identification tab.
2. Set the **Caption** to **iii HR** (where iii=your course assigned ID) and press F1 to see the context-sensitive help for the Caption property.
3. Set the **Hint** to **Human Resources Tutorial**.
4. Ignore the other fields on the tab sheet.



5. Display the Bitmaps and Icons tab.
6. Select the **icon**  for the application.



Note that the icons and bitmaps you see here are shipped with the Framework, but that you can also enrol your own. To see how to do this open the component UF_IB001 in the Visual LANSAs Editor and follow the instructions.

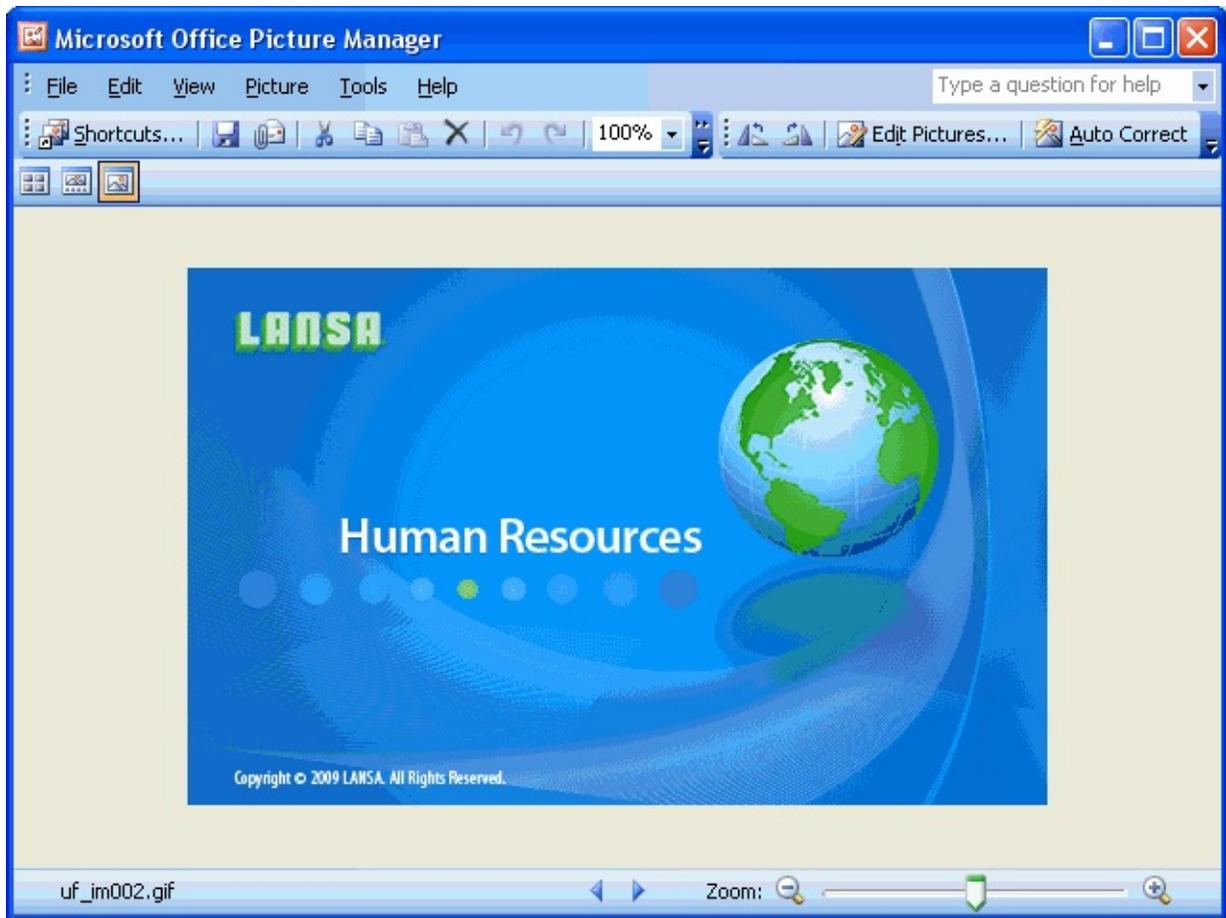
Step 3. Specify Startup Options

In this step you will specify an image that will be displayed when the application is first started up.

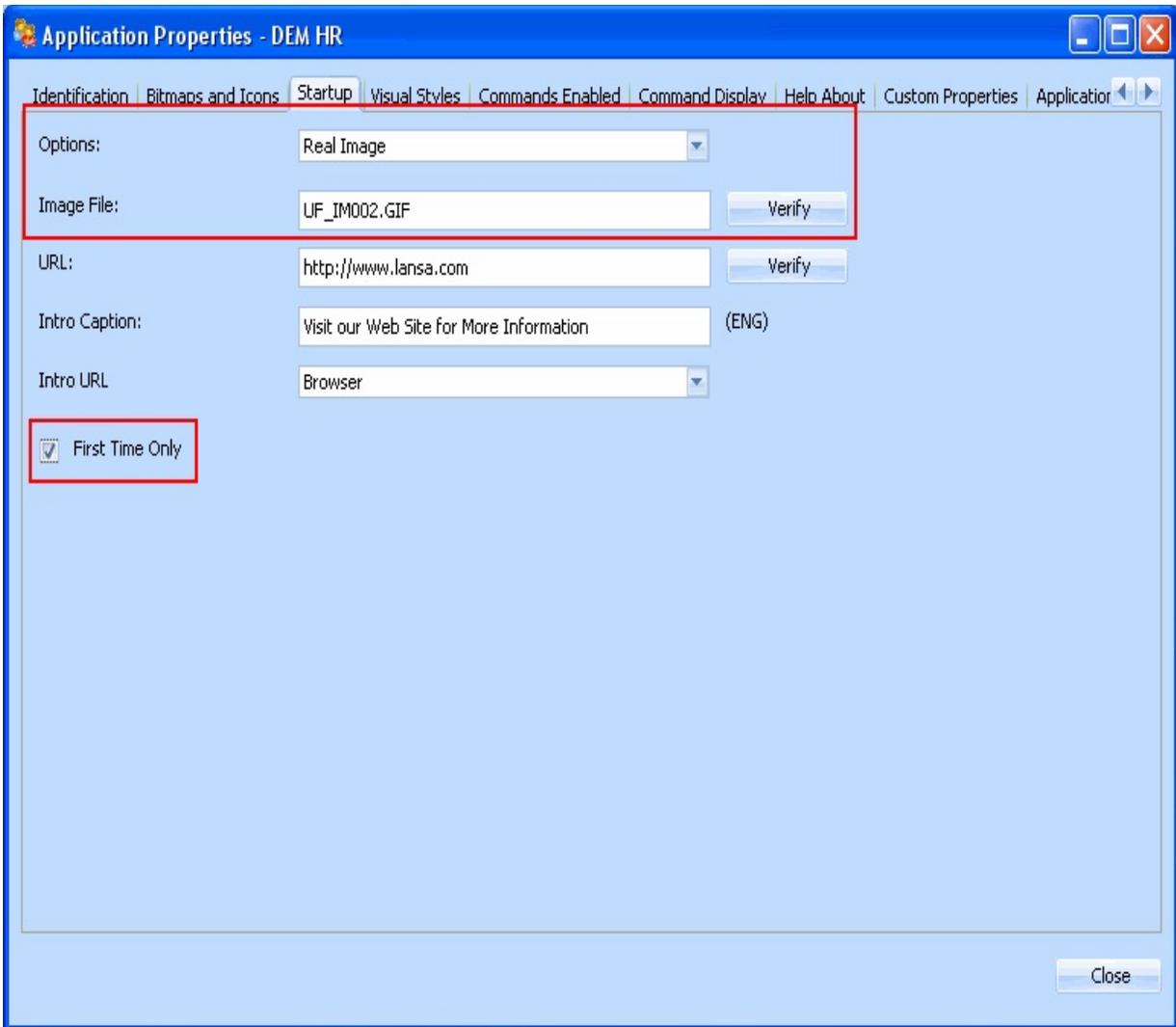
The start up screen allows you to define what the user will see each time they click on your application. It can appear every time they click on it or just the first time they click on it during a single session of using the framework. You can use images or web pages as your splash screen. For example, you might want to direct users to a page on your Intranet depending on which application they access.

1. To change the Framework introduction image, choose the Framework menu, and then the Properties... option. Go to the Startup tab sheet.
2. Set the [Start-up Options](#) to Real Image.
3. Set the [Image File](#) to **UF_IM002.GIF** to display the HR splash screen when the end-user starts up the application.

Click on the Verify button to make sure the image exists. UF_IM002.GIF should appear like this:



4. Select the **First Time Only** option so that the image is only shown once during a session.



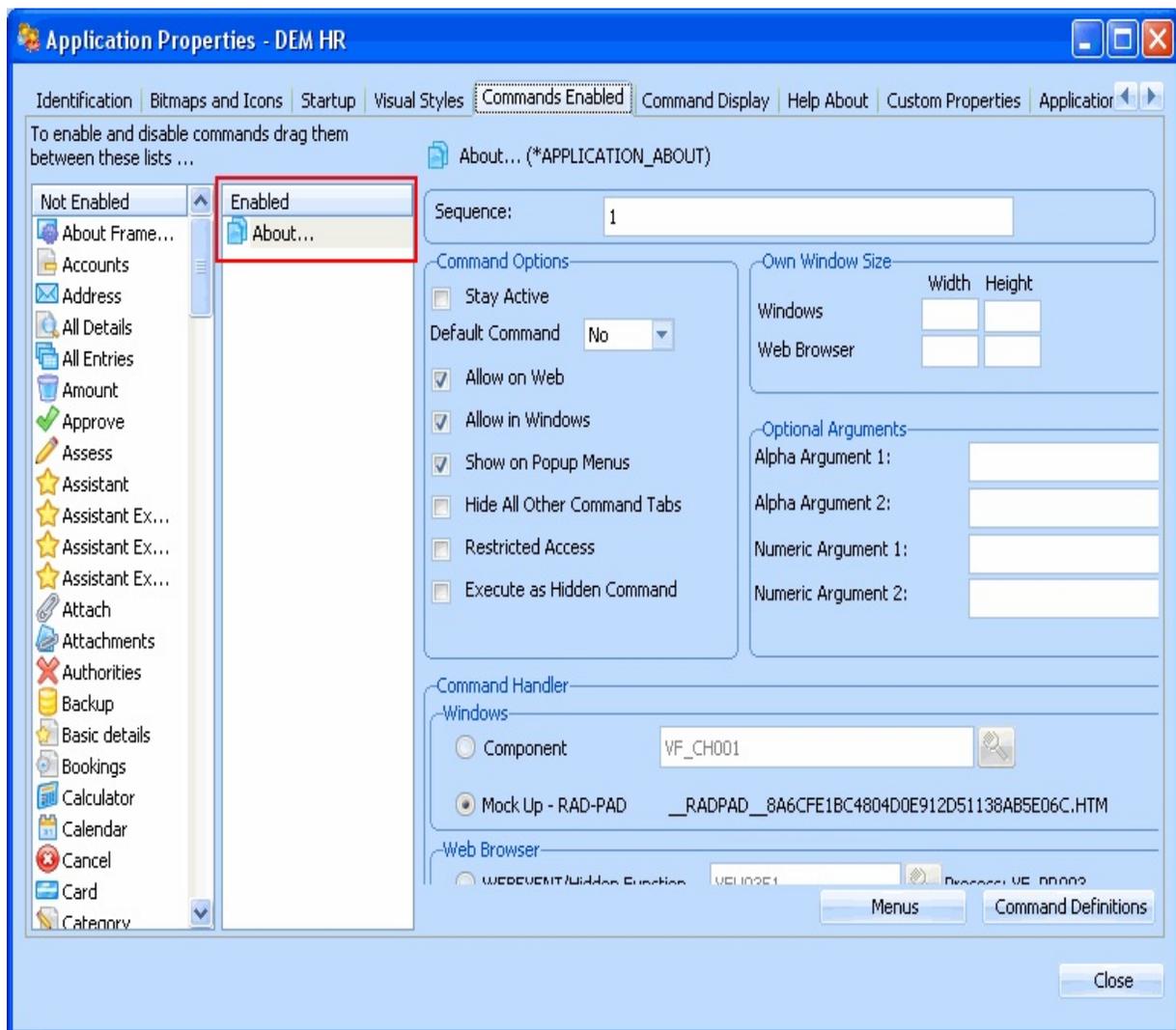
Step 4. View Commands Enabled

In this step, you will review the About command. The About command is enabled by default for all new applications. This command brings up the Help About dialog which provides information about the application.

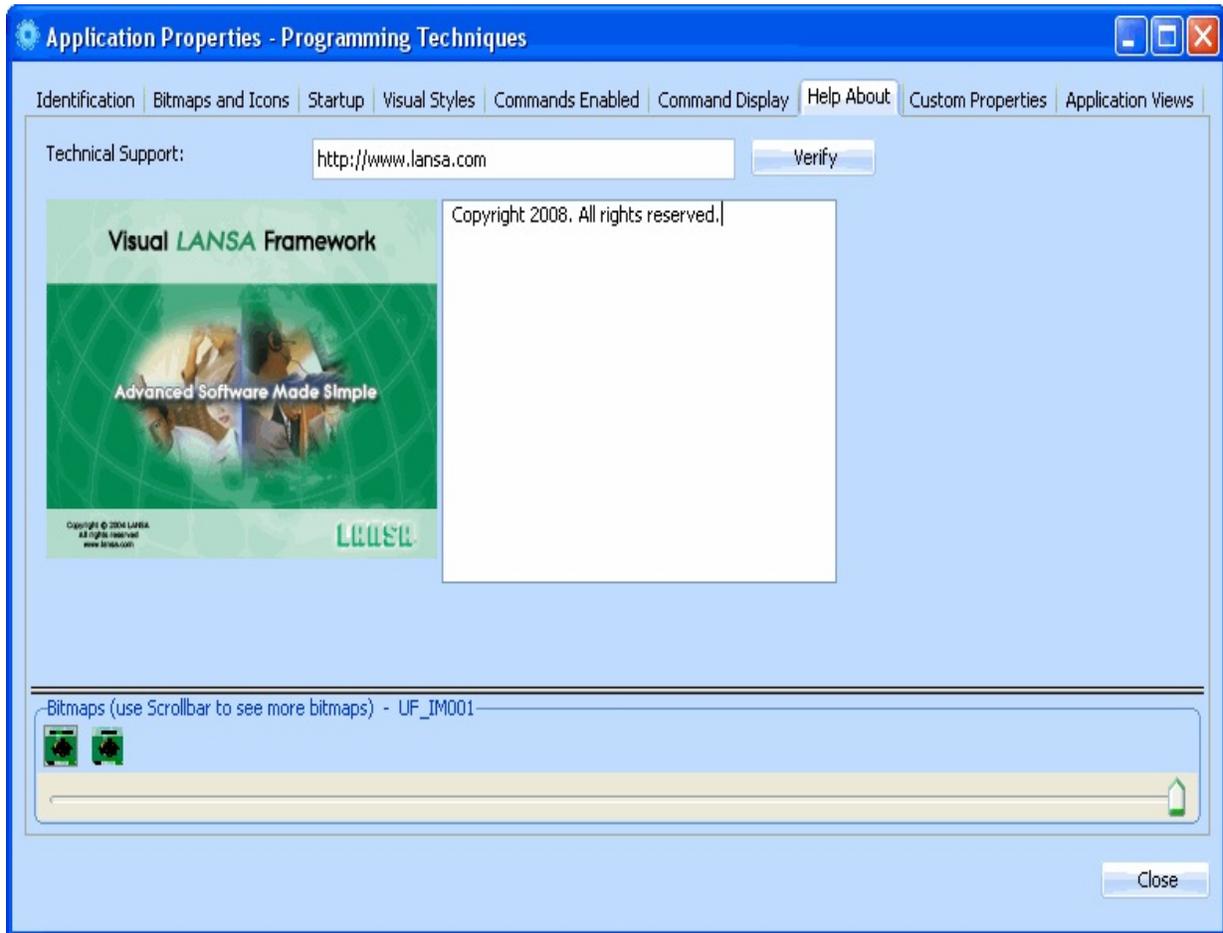
1. Right-click your HR application and choose (Properties...) from the context menu, then display the Commands Enabled tab.

The command list on the left shows all the commands that exist in the Framework. The Enabled column indicates whether the command is enabled for your III HR application.

The About... command is enabled by default for all new applications (it will be added to the pop-up menu of the application).



2. Display the Help About tab.
3. Enter **http://www.lansa.com** for Technical Support (or some other valid Web site).
4. Select the Human Resources bitmap (it is the last one in the bitmap list).
5. Add some text for the copyright in the edit box next to the bitmap.

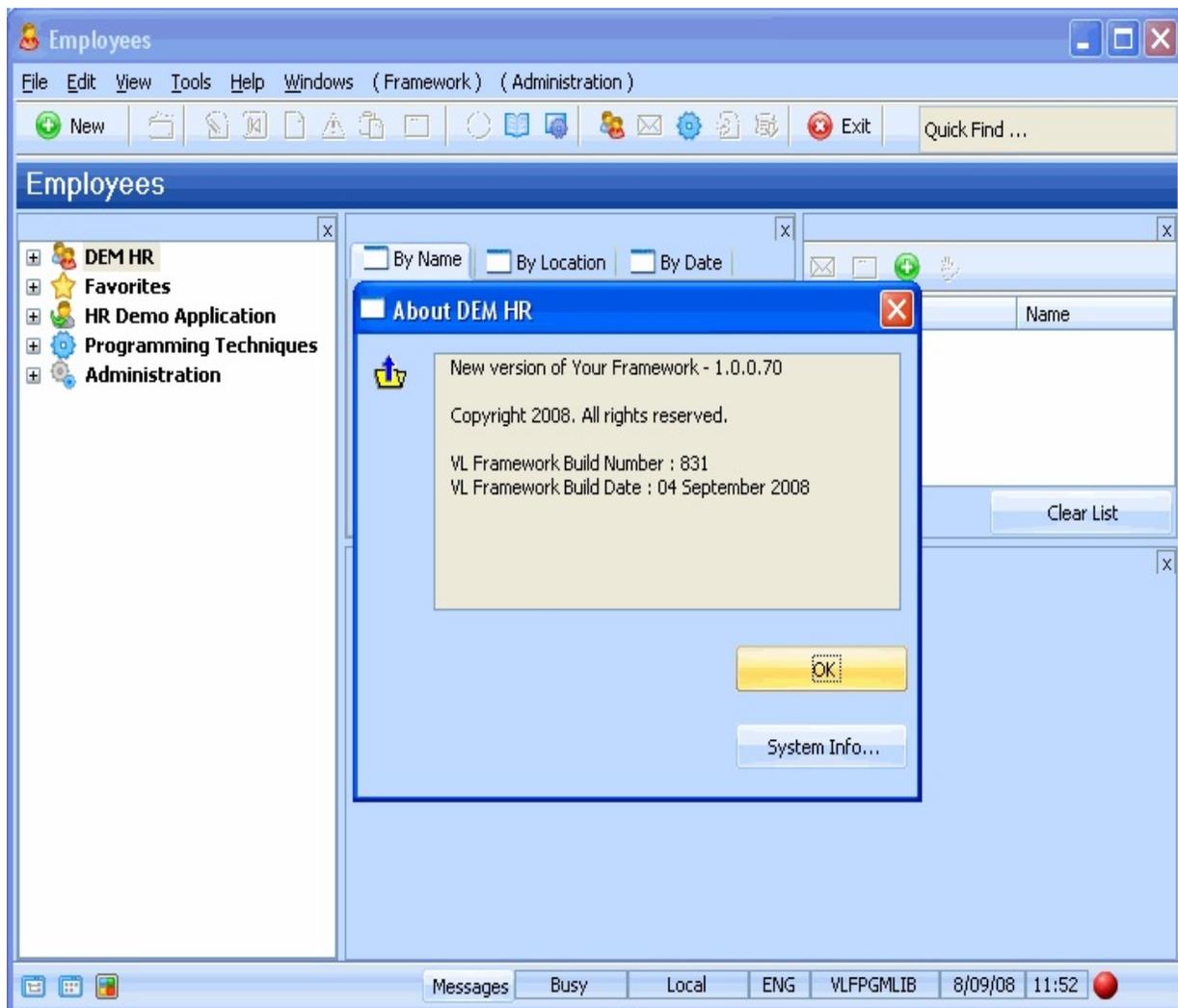


6. Close the Application Properties.

Step 5. Execute the About Command

You have now finished defining your application. In this step, you will try executing the application and the only command (About...) that is currently enabled for it. When commands are enabled they can often be executed from different places.

1. Select your iii HR application and right-click.
2. Select the About... command from the pop-up menu. This displays your Iii HR application's help about information:



3. Click OK.
4. Now Select Help and then About.... from the main menu bar. Again, your III

HR application's help about information is presented.

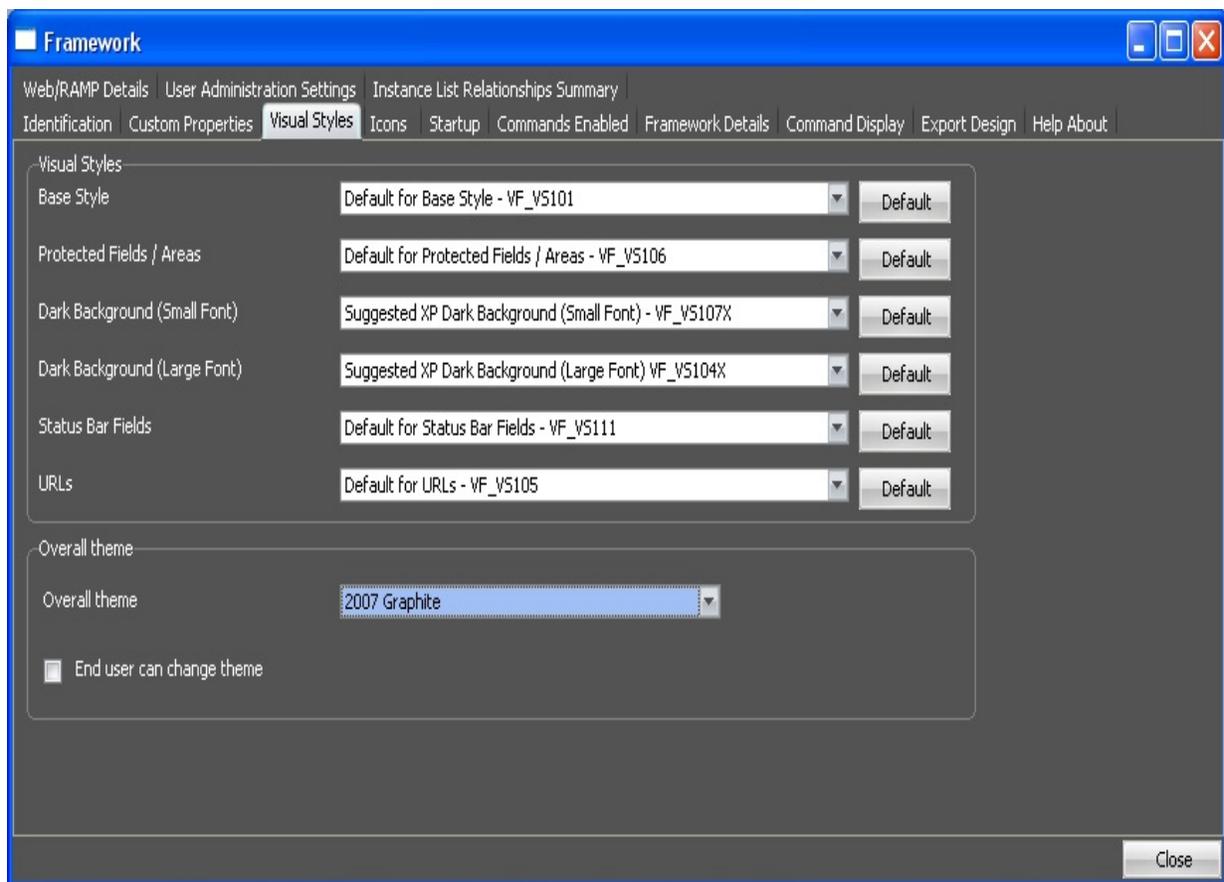
5. Click on the Technical Support... button to display the web page you specified.
6. Click OK to close the dialog box.

Step 6. View Overall Themes

This step is only applicable to EPC831 version of the Framework with Visual LANSA SP5 or later.

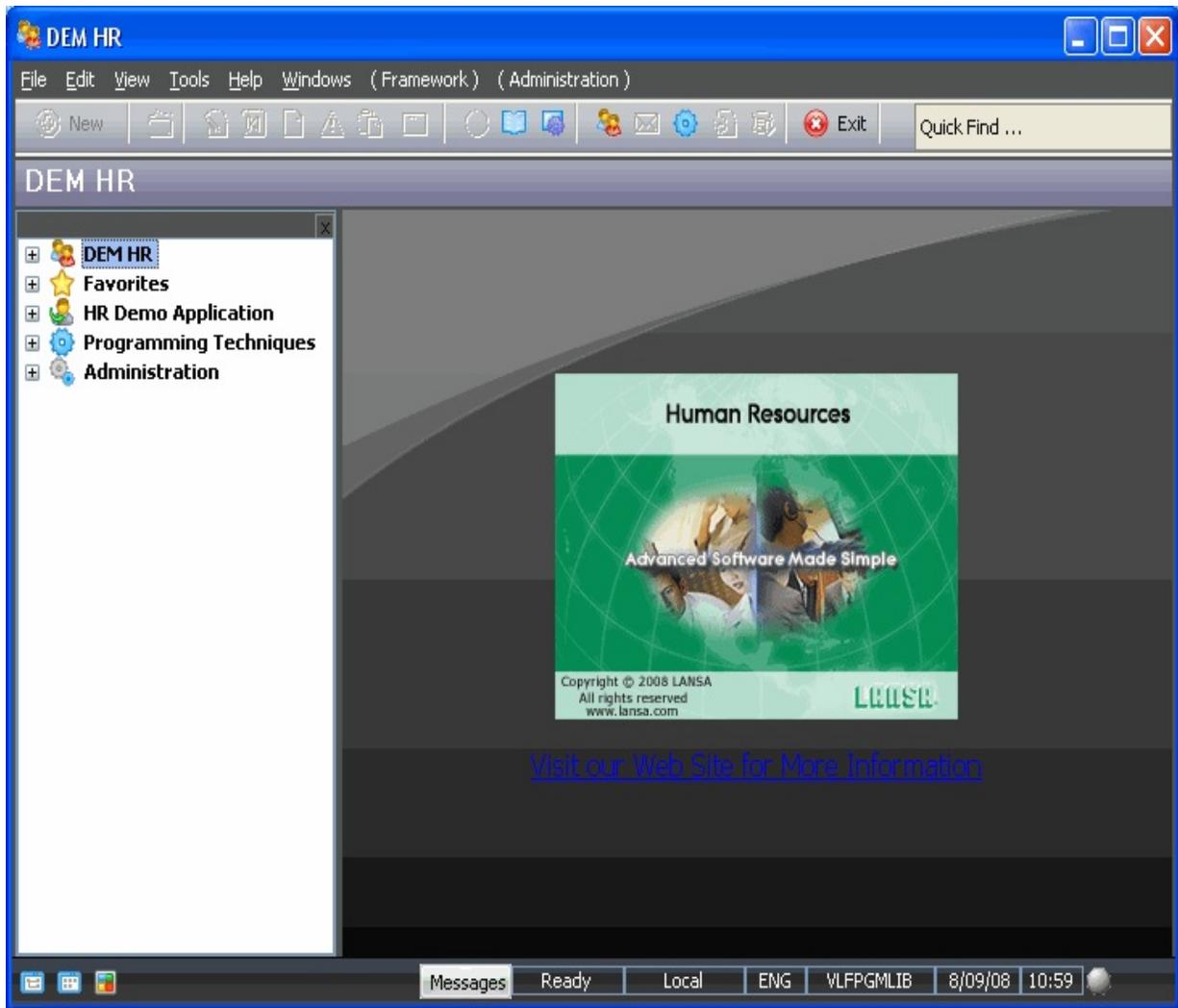
In this step you find out how to change the visual theme of your application.

1. Display the Framework properties.
2. Display the Visual Styles tab.
3. Make sure the option End user can change theme is not selected.
3. Change the Overall Theme to 2007 Graphite:



When a theme is selected the appearance of the entire framework changes to the new theme immediately, including your own command handlers and filters.

4. Close the Framework properties window and click on the iii HR application to see how it has changed.



5. Change the theme back to 2007 Blue in the Framework properties.

Summary

Important Observations

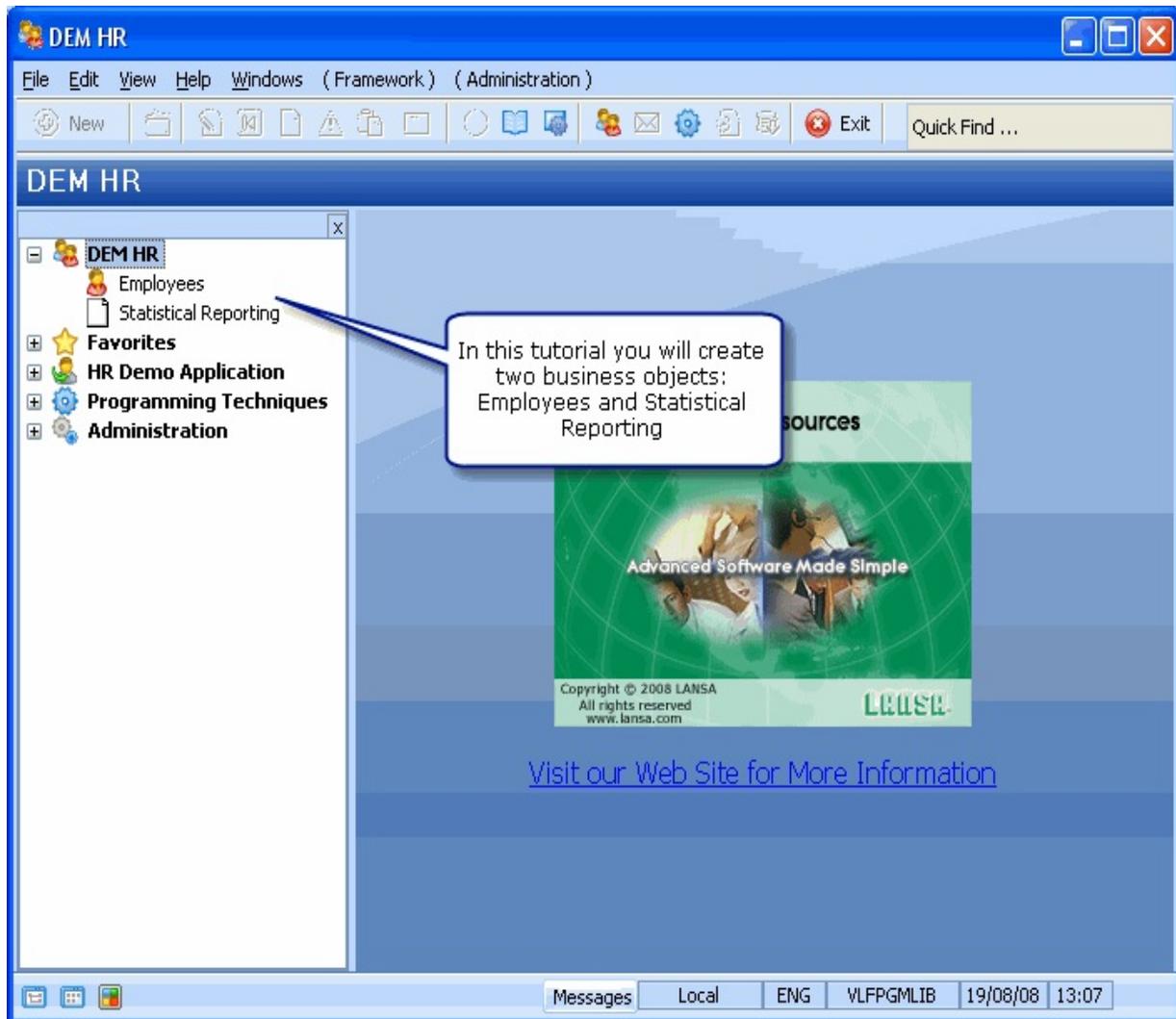
- To create new applications with the Visual LANSAs Framework, you simply set the application properties. You do not have to write any code.
- The About... command has a Built-In command handler. For many other command handlers you will need to specify the name of a command handler and other options.
- In the preceding steps you created the application manually. You could also have used the Instant Prototyping Assistant (IPA) to create the application for you. The next tutorial shows how to use the IPA.

What I Should Know

- How to create an application.
- How to set application properties.
- What commands are and how to enable them (more will come later).
- What the purpose of an application is. You may want to review [Application](#).
- What an application bar is. You may want to review [Framework Window](#).

VLF002 - Defining Your Business Objects Objective

- Learn how to use the Instant Prototyping Assistant.
- Learn how to define business objects in your application.
- To add the Employees and Statistical Reporting business objects to your application.



To achieve this objective, you will complete the following steps:

- [Step 1. Decide on the Business Objects](#)
- [Step 2. Create the Business Objects](#)
- [Step 3. Specify Business Object Commands](#)
- [Step 4. Add the Business Objects to the Application](#)

- [Step 5. Specify Business Object Icons](#)
- [Summary](#)

Before You Begin

You may wish to review:

- [Business Objects](#) in [Key Concepts](#).

In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)

Step 1. Decide on the Business Objects

You need to consider carefully what business objects your application will contain. Your decision should be based on an analysis of the tasks of the users of your application. The way the data is stored (i.e. the database structure) is not important.

Your application may have different kinds of users with different needs.

Usually there are various ways of structuring the application. A good solution is logical, intuitive and avoids duplication.

In this tutorial you will create these two business objects for the HR application:

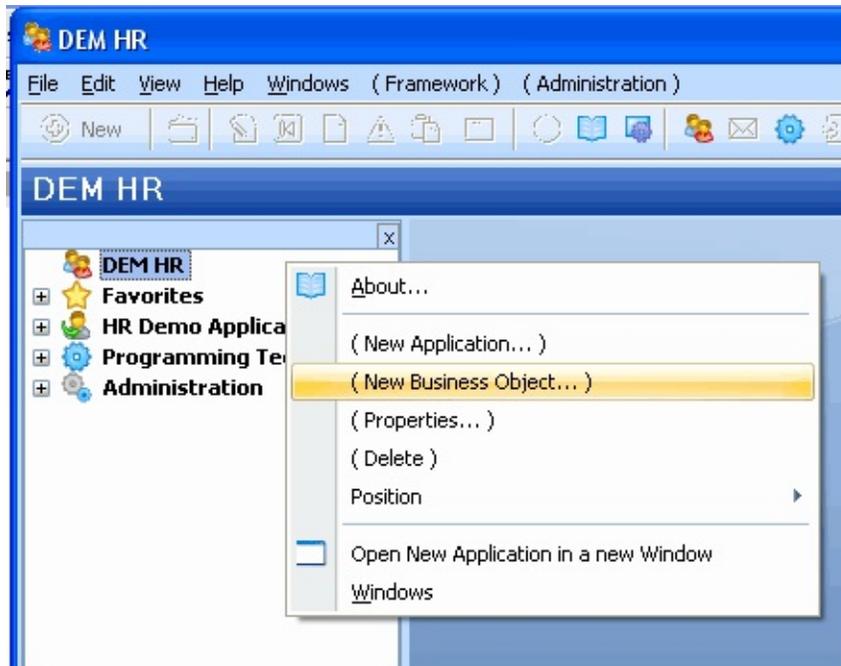
- Employees
- Statistical Reporting

For a different, more complex solution have a look at [Demonstration Application](#).

Step 2. Create the Business Objects

In this step, you will define the business objects.

1. Select the III HR application.
2. Right-click the application to bring up the pop-up menu and choose the New Business Object... option.



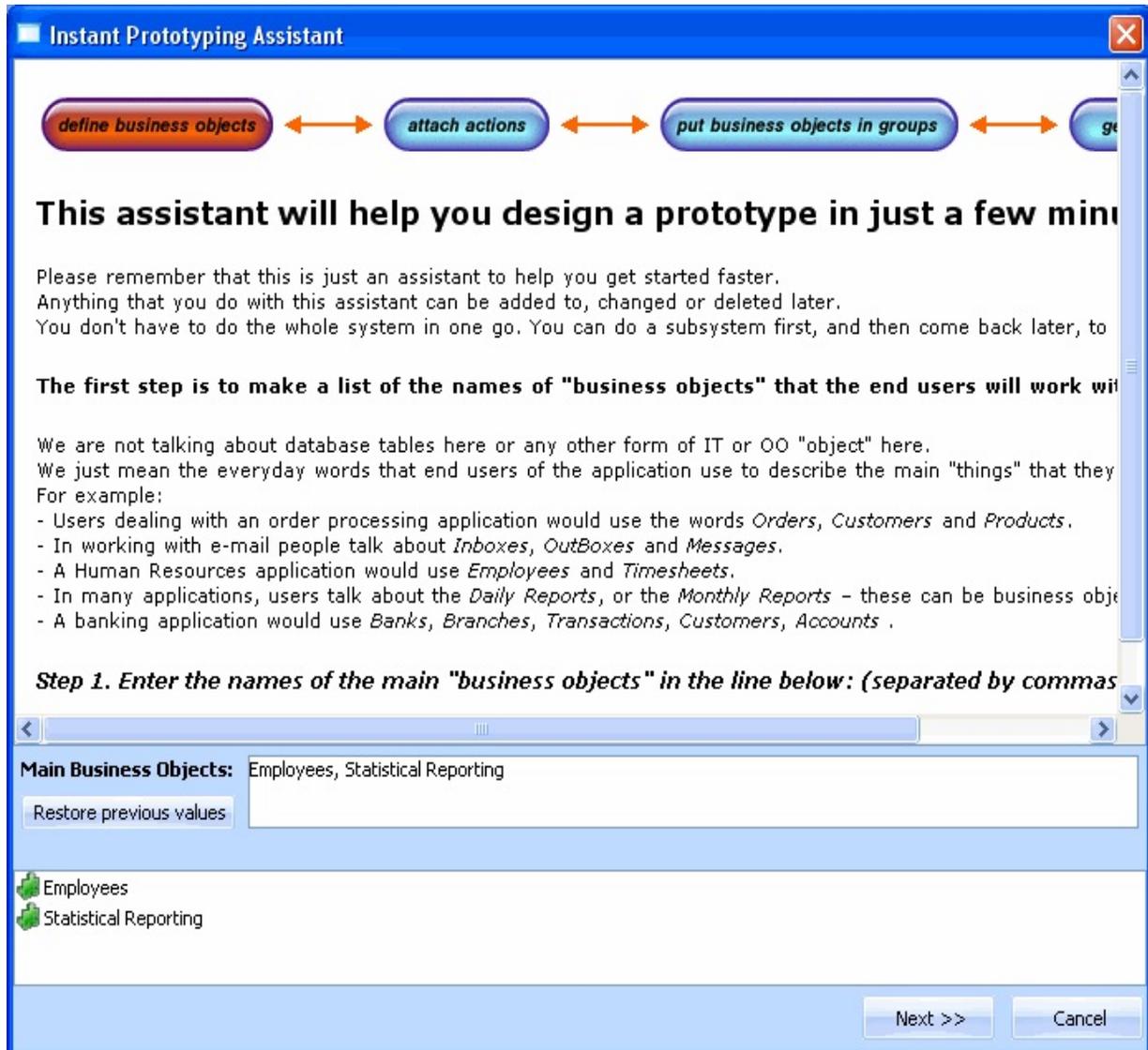
A dialog asking if you want to use the Instant Prototyping Assistant is displayed:



The Instant Prototyping Assistant allows you to define the entire structure of your application including all the applications, business objects within those applications and the commands that are used by each business object. You can always manually add everything, but when you are starting your prototype this

method provides results much more quickly.

3. Click on the Yes button. The Instant Prototyping Assistant window is displayed.
4. Type in Employees and Statistical Reporting as the Main Business Objects that will be part of your HR application. Notice that each business object is separated by a comma.

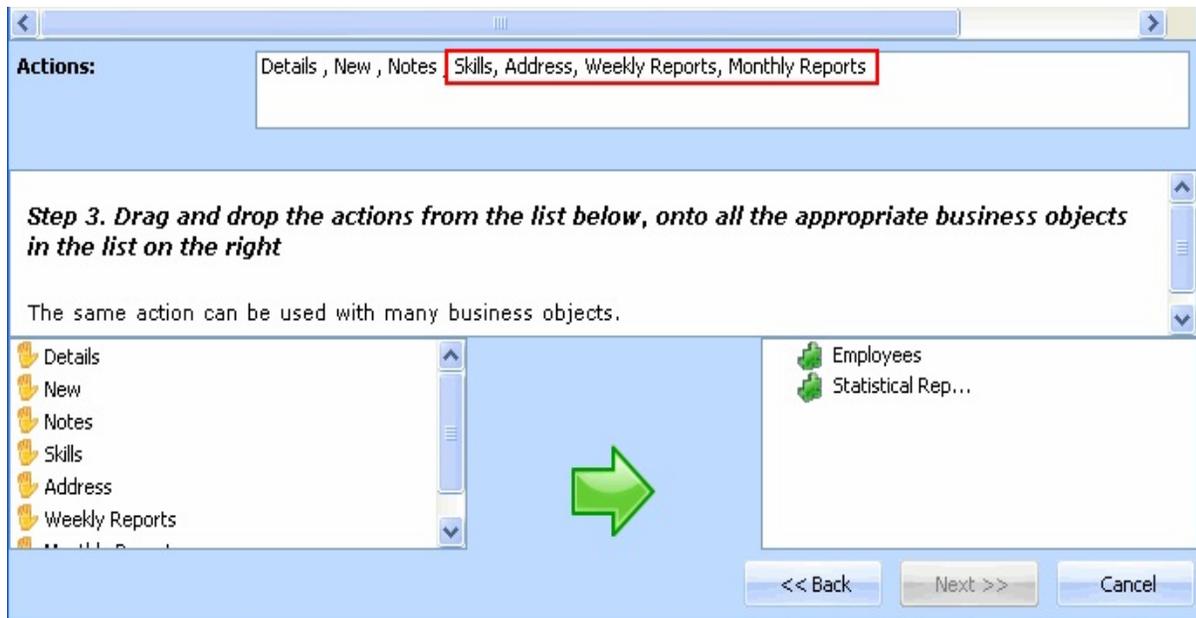


5. Click on the Next button.

Step 3. Specify Business Object Commands

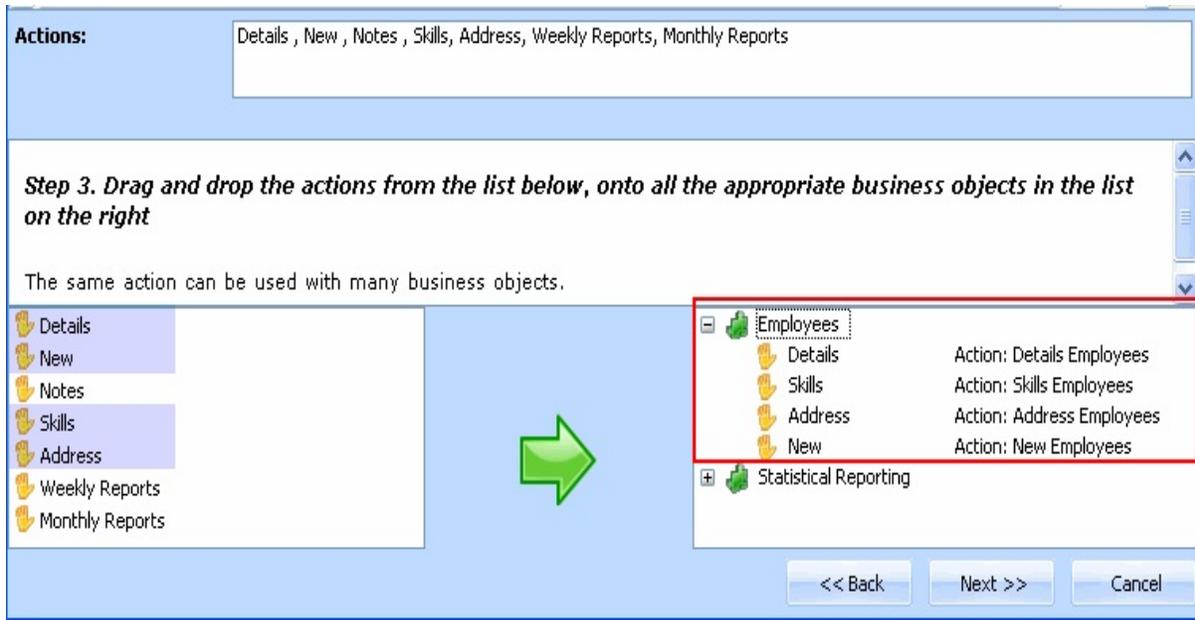
In this step you will add commands for the business objects.

1. In the list of Actions type in Skills, Address, Weekly Reports, Monthly Reports after the existing actions. The commands need to be separated with a comma.

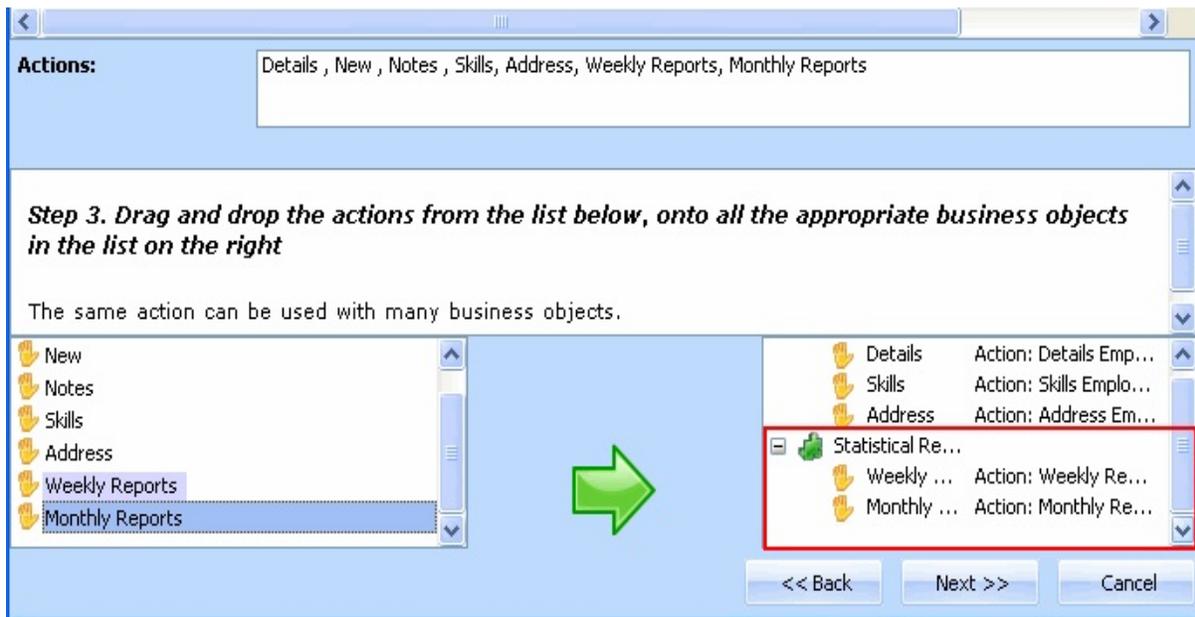


Next associate the commands with the business objects:

2. Drag and drop the Details, New, Skills and Address commands to the Employees business object:



3. Drag and drop the Weekly Reports and Monthly Reports commands to the Statistical Reporting business object.



4. Click on the Next button.

Step 4. Add the Business Objects to the Application

In this step you associate the business objects with the application.

1. Select the Employees and the Statistical Reporting business objects.

Applications: Programming Techniques , Favorites , HR Demo Application , Administration , DEM HR

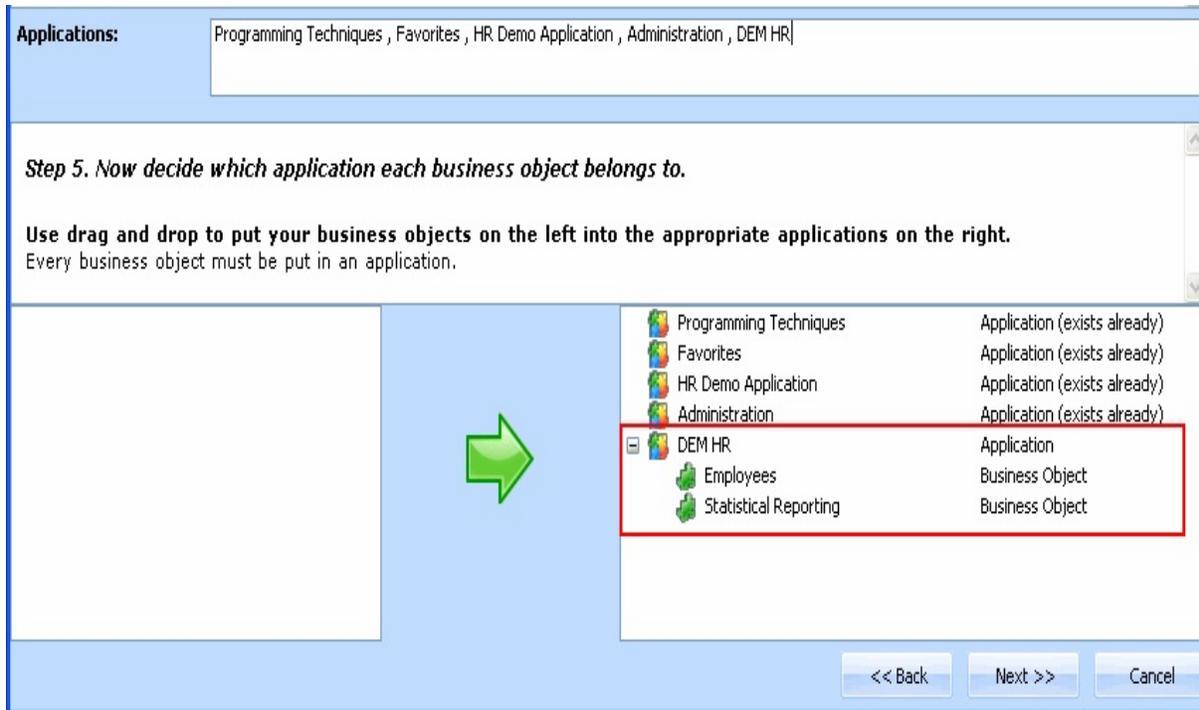
Step 5. Now decide which application each business object belongs to.

Use drag and drop to put your business objects on the left into the appropriate applications on the right.
Every business object must be put in an application.

Employees	Programming Techniques	Application (exists already)
Statistical Reporting	Favorites	Application (exists already)
	HR Demo Application	Application (exists already)
	Administration	Application (exists already)
	DEM HR	Application

<< Back Next >> Cancel

2. Drag them to the iii HR application:

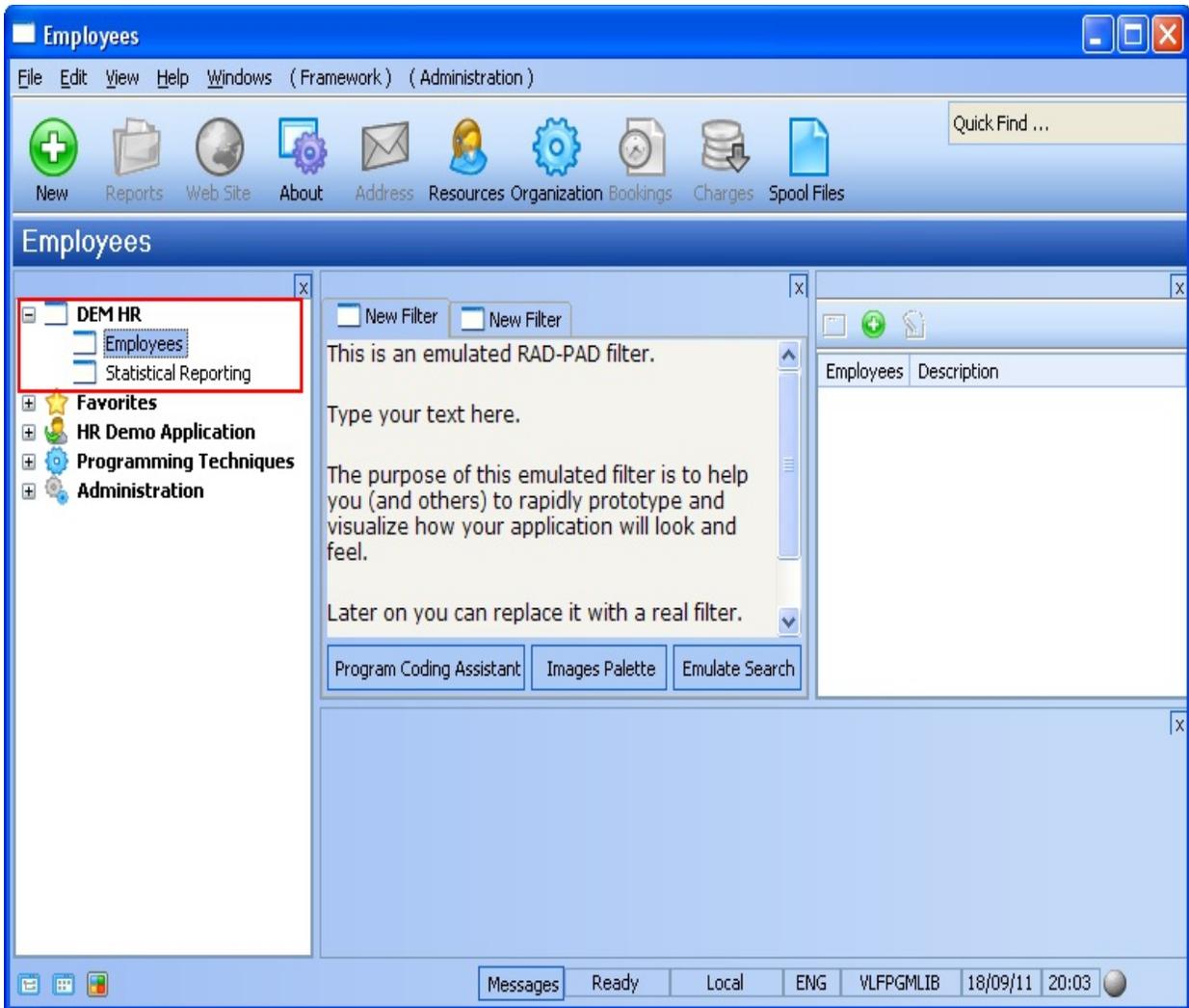


Note that instead of manually defining your application as you did in [VLF001 - Defining Your HR Application](#), you could have added it on this screen to the list of existing applications. It is important to realize that you can prototype an entire system of many applications using the Instant Prototyping Assistant.

3. Click Next. A summary of your application prototype is shown.

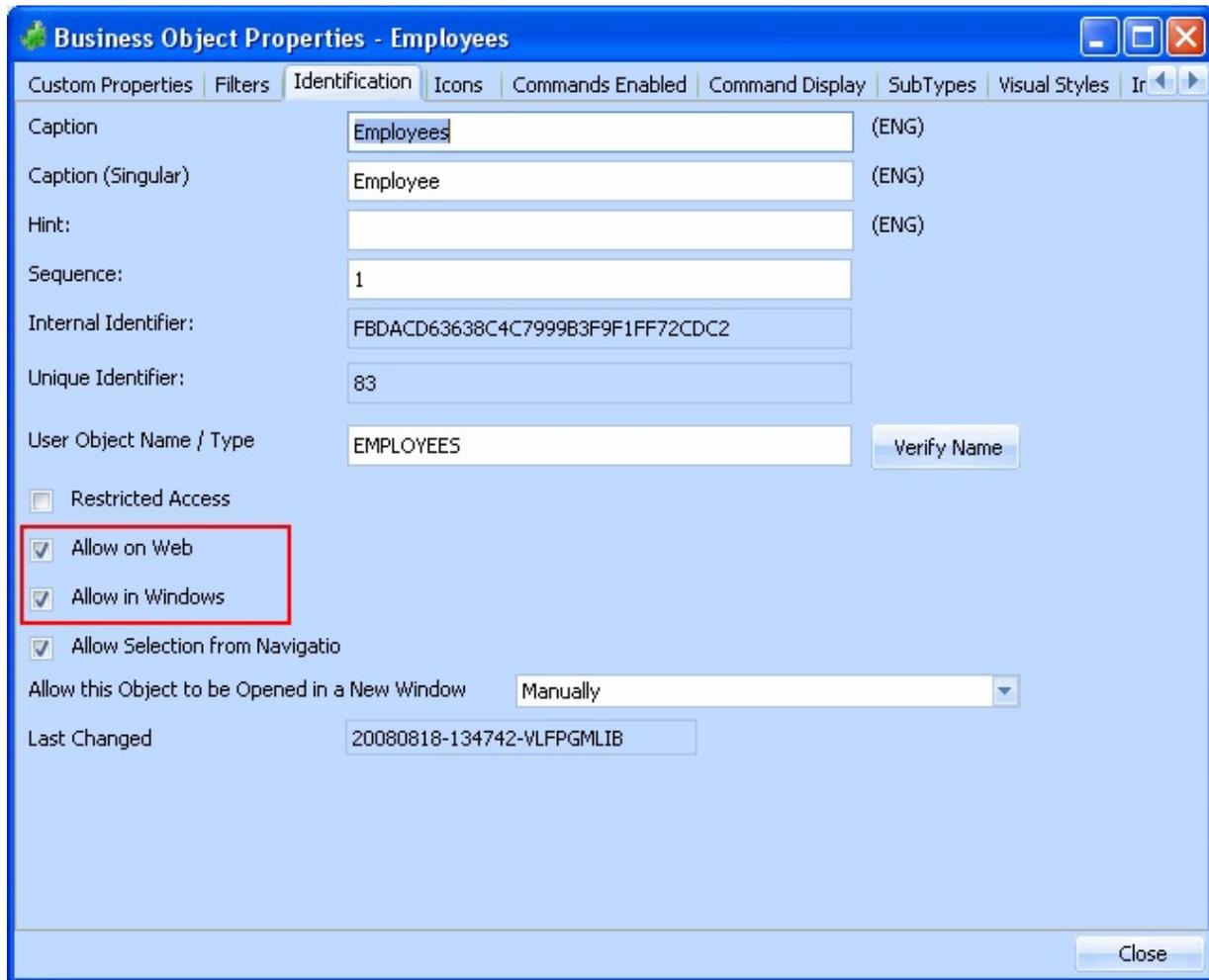
4. Click Finish to create the prototype.

You will now see your application and business objects:



Step 5. Specify Business Object Icons

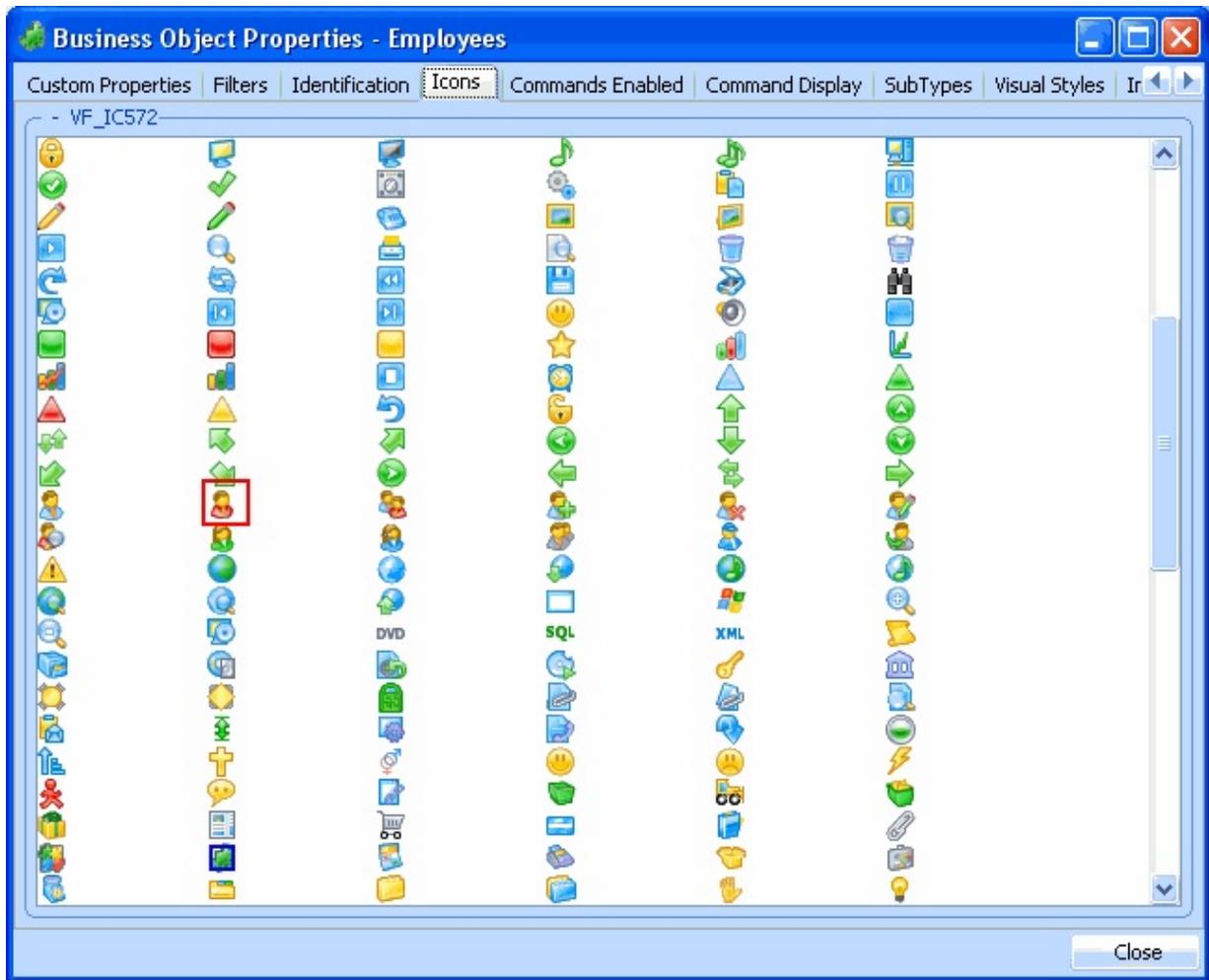
1. Double-click the Employees business object to display its properties. Note that the Allow on Web and Allow in Windows options are selected by default. These options control in which environment the business object can be used.



The screenshot shows the 'Business Object Properties - Employees' dialog box. The 'Identification' tab is selected. The 'Caption' is 'Employees' (ENG), 'Caption (Singular)' is 'Employee' (ENG), and 'Hint:' is empty (ENG). The 'Sequence' is '1', 'Internal Identifier' is 'FBDACD63638C4C7999B3F9F1FF72CDC2', and 'Unique Identifier' is '83'. The 'User Object Name / Type' is 'EMPLOYEES' with a 'Verify Name' button. The 'Restricted Access' checkbox is unchecked. The 'Allow on Web' and 'Allow in Windows' checkboxes are checked and highlighted with a red box. The 'Allow Selection from Navigation' checkbox is checked. The 'Allow this Object to be Opened in a New Window' dropdown is set to 'Manually'. The 'Last Changed' field shows '20080818-134742-VLFPGMLIB'. A 'Close' button is at the bottom right.

Whenever you create an application, business object, filter or command handler, you can specify in which environment it is to be used for simply by checking these options.

2. Display the Icons tab.
3. Select an [icon](#) for the Employees business object.



4. Without closing the properties tab folder, click on the Statistical Reporting business object in the iii HR application to set its icon.

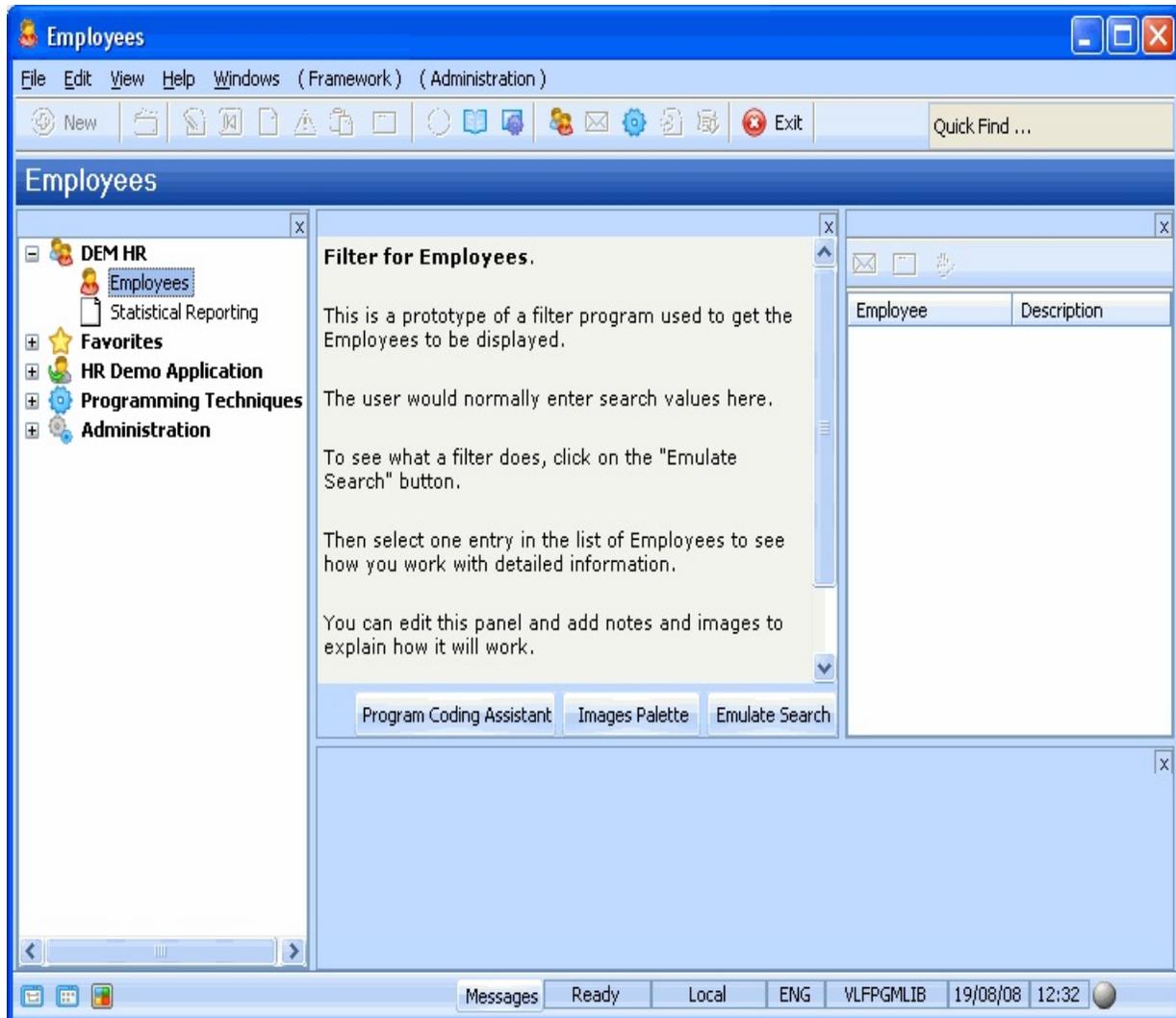


Step 6. View the Business Objects

In this step, you will have a look at what your business objects look like now.

1. Close the Business Object Properties tab folder.
2. Click on the Employees business object.

The Employees business object is displayed in your application:



In the window you can also now see a mock up [Filter](#) and next to it an [Instance List](#) . However, your definitions of the Employees is not yet complete. You will learn more about filters and instance list in [VLF003 - Prototyping Your Filters](#).

Summary

Important Observations

- Applications may contain many different business objects. Business objects are the objects that an end-user works with in the application.

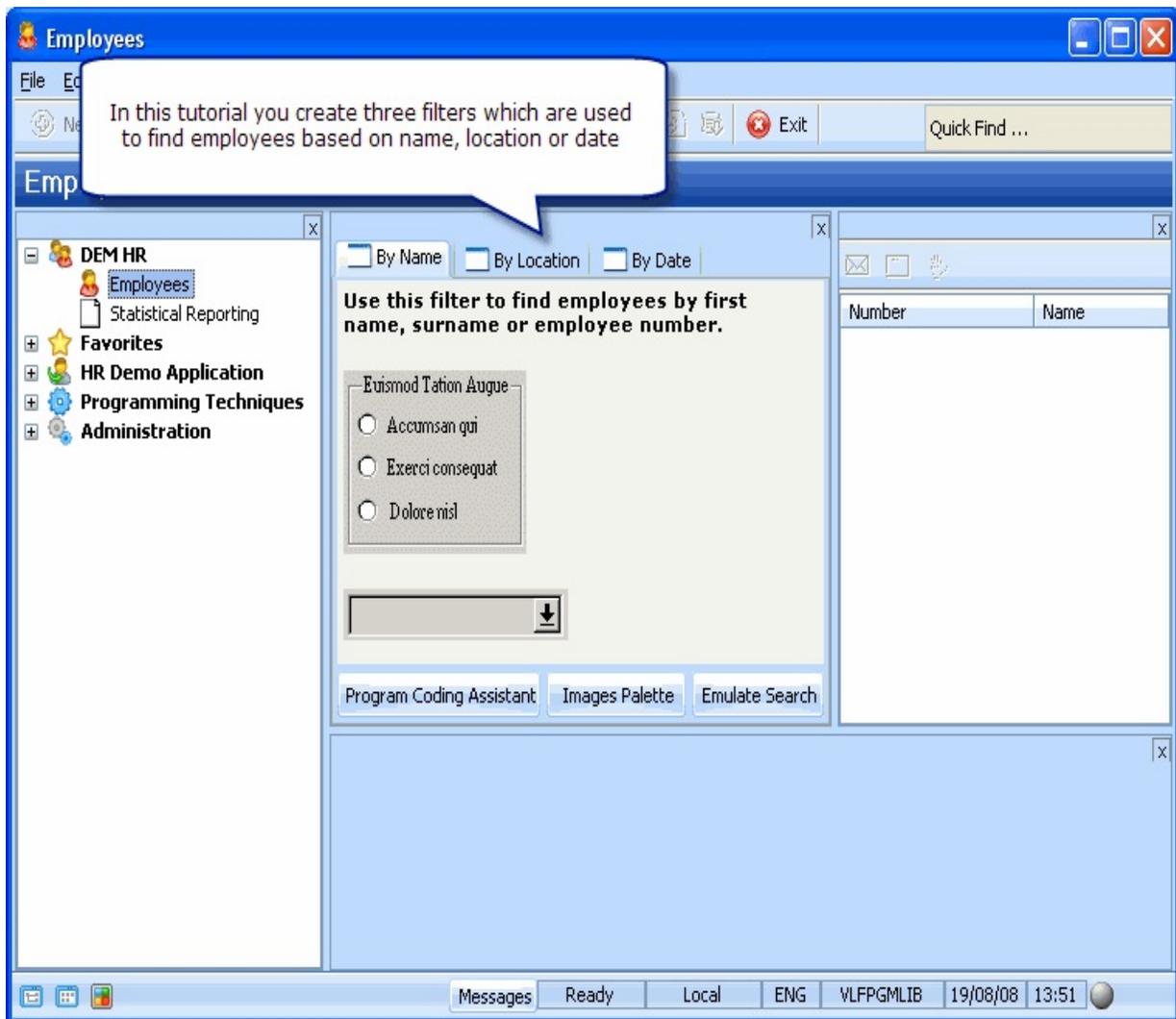
What I Should Know

- How to create business objects.
- How to define business object properties.
- What a business object is. You may want to review [Business Object](#).

VLF003 - Prototyping Your Filters

Objective

- Learn how to create filters which populate the [Instance List](#) with selected items.
- To add filters to search for employees by name, by location and by date.



To achieve this objective, you will complete the following steps:

- [Step 1. Define By Name Filter for Employees](#)
- [Step 2. Prototype the Instance List](#)
- [Step 3. Prototype the Filter Designs for Employees](#)
- [Step 4. Filters for Statistical Reporting Business Object](#)
- [Step 5. View the Filters](#)

- [Summary](#)

Before You Begin

You may wish to review:

- [Filters](#) in [Key Concepts](#).

In order to complete this tutorial, you must have completed the following:

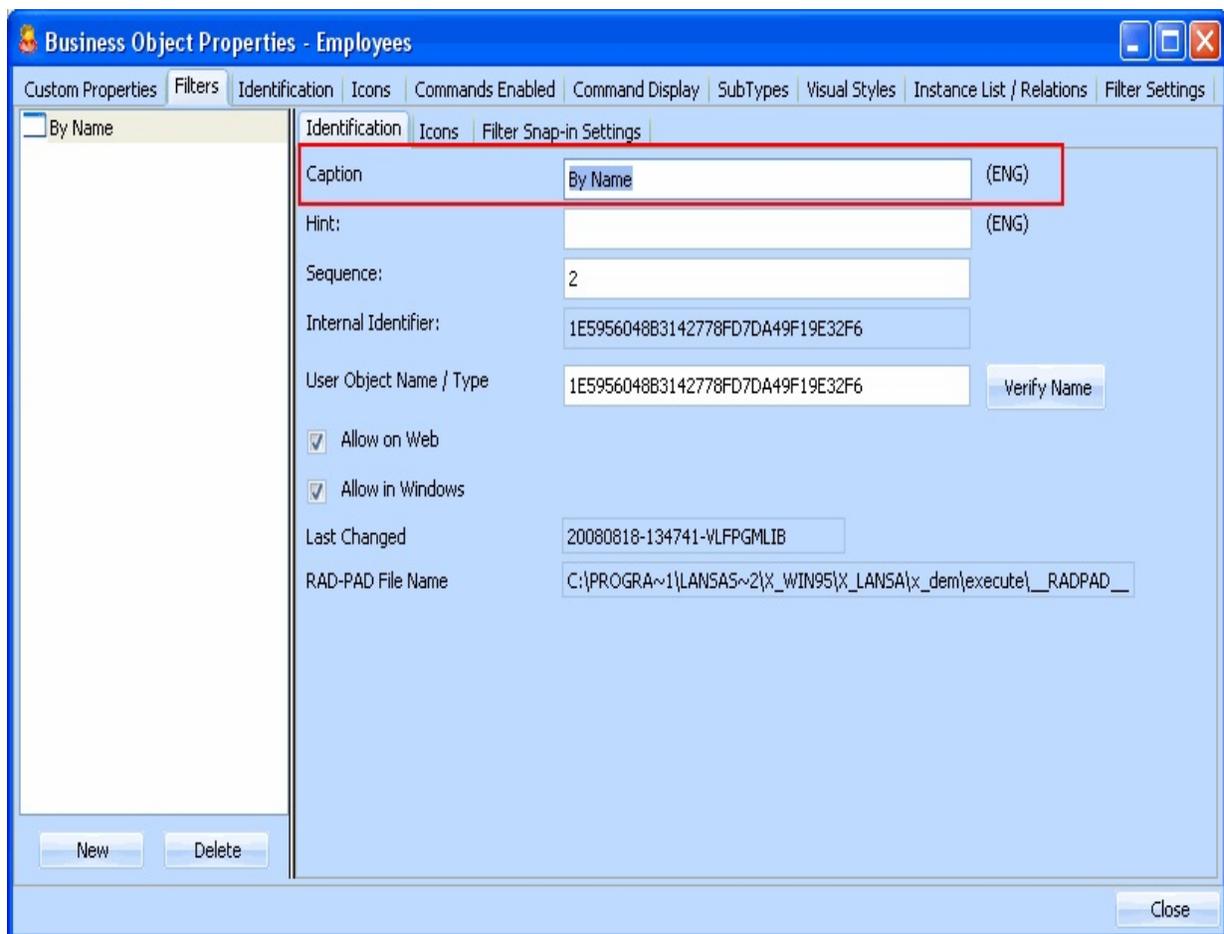
- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)

Step 1. Define By Name Filter for Employees

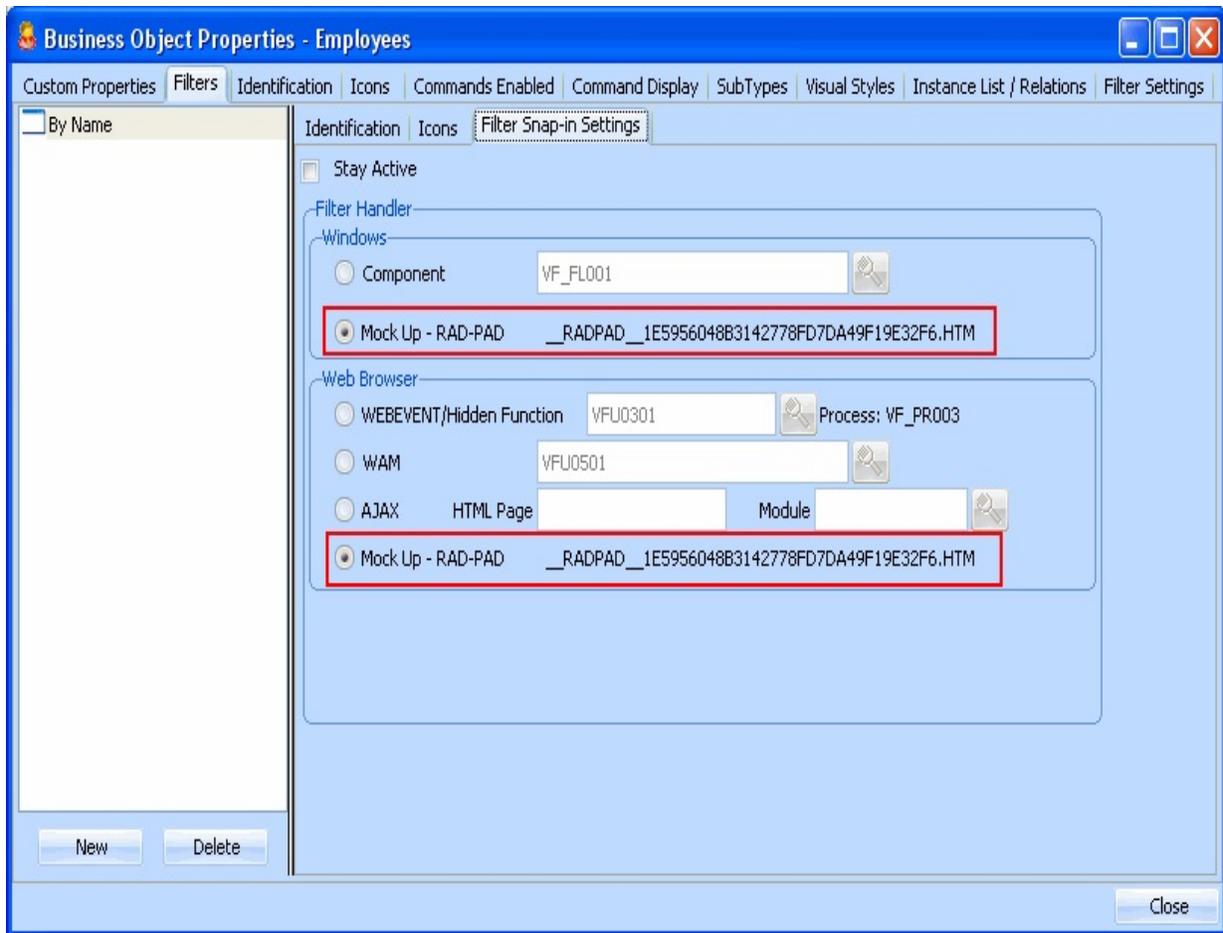
1. Select your Employees business object.
2. Right-click to display the pop-up menu and select (Properties...).
3. Select the Filters tab. A new filter is already defined for Employees by default.

(Note that the sequence number of new filters depends on what other objects are defined in your system. The numbering does not matter.)

4. Set the **By Name** of the filter to **by Name**.



5. Display the Filter Snap-in Settings tab.
6. Make sure **Mock Up RAD-PAD** filter type is selected. You want to use the RAD-PAD filters so that you can enter text and pictures in the prototype.



7. Create a new filter by clicking on the New button and set the **Caption** of your second filter to **by Location**.
8. Create another new filter by clicking the New button in and set the **Caption** of your third filter to **by Date**.
10. Do not change any of other default values for the filters.

Step 2. Prototype the Instance List

An instance list is a list of results returned from your filter. You can define one or more columns that are visible in your instance list. There are four visible types of columns:

- The VISUALID1 and VISUALID 2 columns are alphanumeric.
- The ACOLUMN fields are the additional alphanumeric data that you can add to the list.
- The NCOLUMN fields are the additional numeric data that can be added to the list
- The DCOLUMN fields are the additional date or datetime data that can be added to the list

In this step, you will change the column headings in the instance list to make the prototype more realistic.

1. In the Employee business object properties select the Instance List / Relations tab.
2. Set the first Column Caption to be **Number**.
Set the column width to be 25%.
3. Set the second column heading to be **Name**.

Business Object Properties - Employees

Save and Restore Instance Lists
 Allow multiple selections
 Allow Instance List to be sent to MS-Excel File Prefix to be used for MS-Excel: Spreadsheet_ (ENG)

Instance List Tool Bar Location: Top
 Instance List Tool Bar Text Location: <None>
 Instance List Tool Bar Height or Width: 24
 Snap in Instance List Browser:

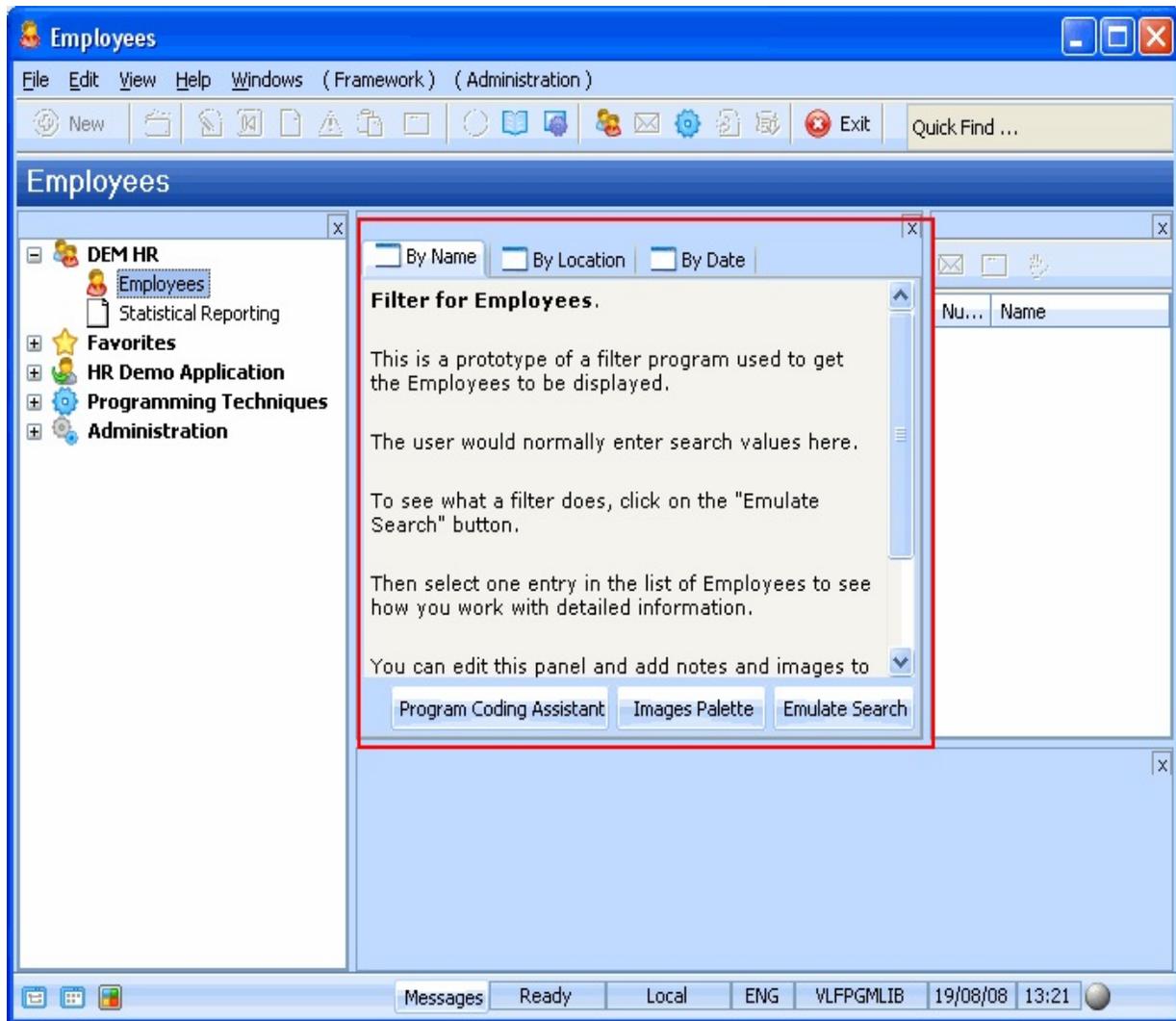
Sequence	Type	Caption	Width % (Total 25%)	Decimals	Edit Code	Date/Time Output Format	UTC Conversion
10	VISUALID1	Number	25		Default	SYSFMT8	Local -> Local
20	VISUALID2	Name			Default	SYSFMT8	Local -> Local
	ACOLUMN1				Default	SYSFMT8	Local -> Local
	ACOLUMN2				Default	SYSFMT8	Local -> Local
	ACOLUMN3				Default	SYSFMT8	Local -> Local
	ACOLUMN4				Default	SYSFMT8	Local -> Local
	ACOLUMN5				Default	SYSFMT8	Local -> Local
	ACOLUMN6				Default	SYSFMT8	Local -> Local

Close

Step 3. Prototype the Filter Designs for Employees

In this step, you will prototype the contents of the filters.

1. Close the Business Object Properties dialog box. You can now see your three filters (By Location, By Name and By Date) for the Employees business object:

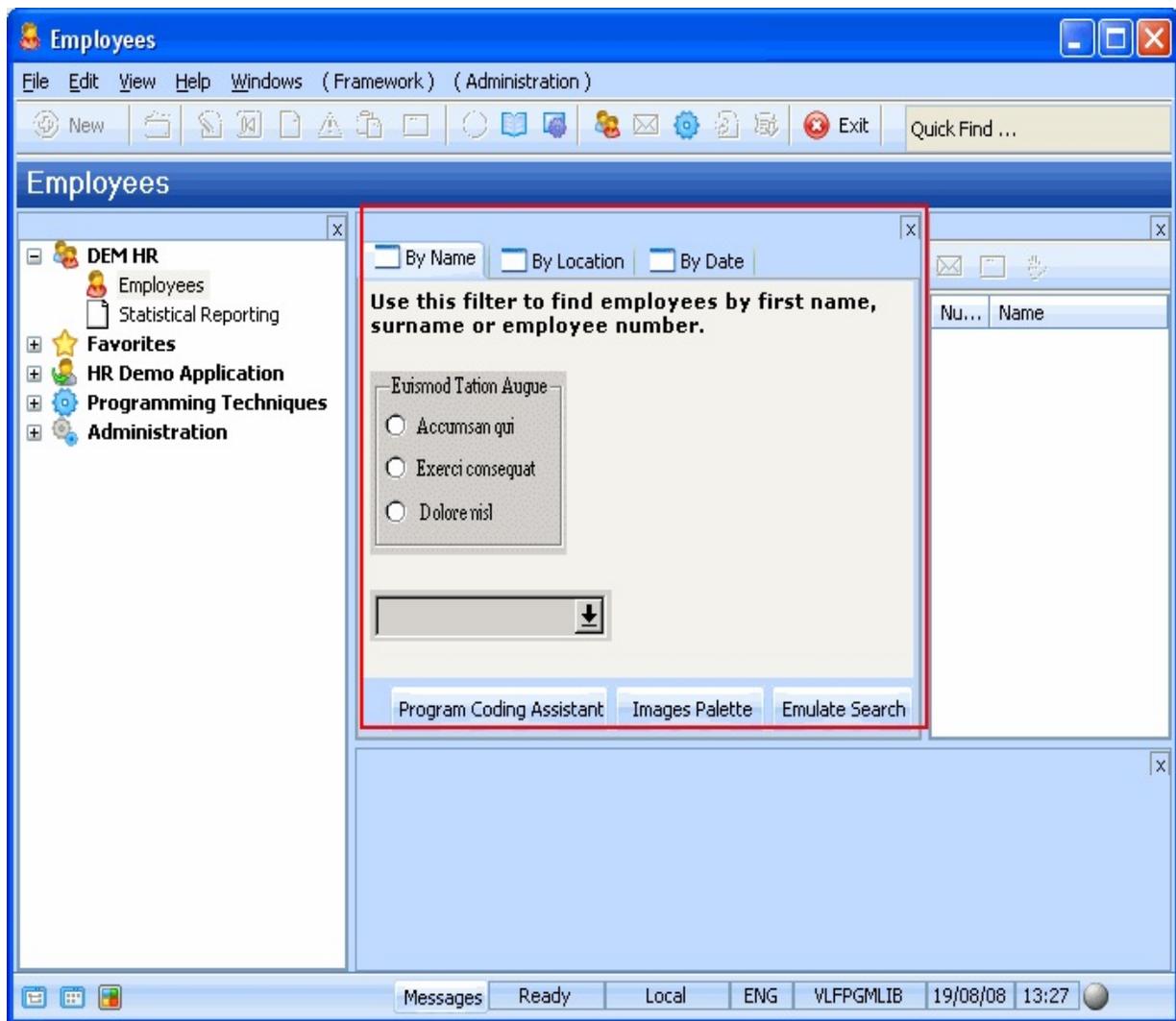


If you have a close look you can see that the [Mock Up RAD-PAD](#) filter has the standard text which you can select and delete and then type your own text and paste in pictures.

Before you start using the RAD-PADs it is important you understand:

- They are meant for quick notes with pictures, not for formal screen designs

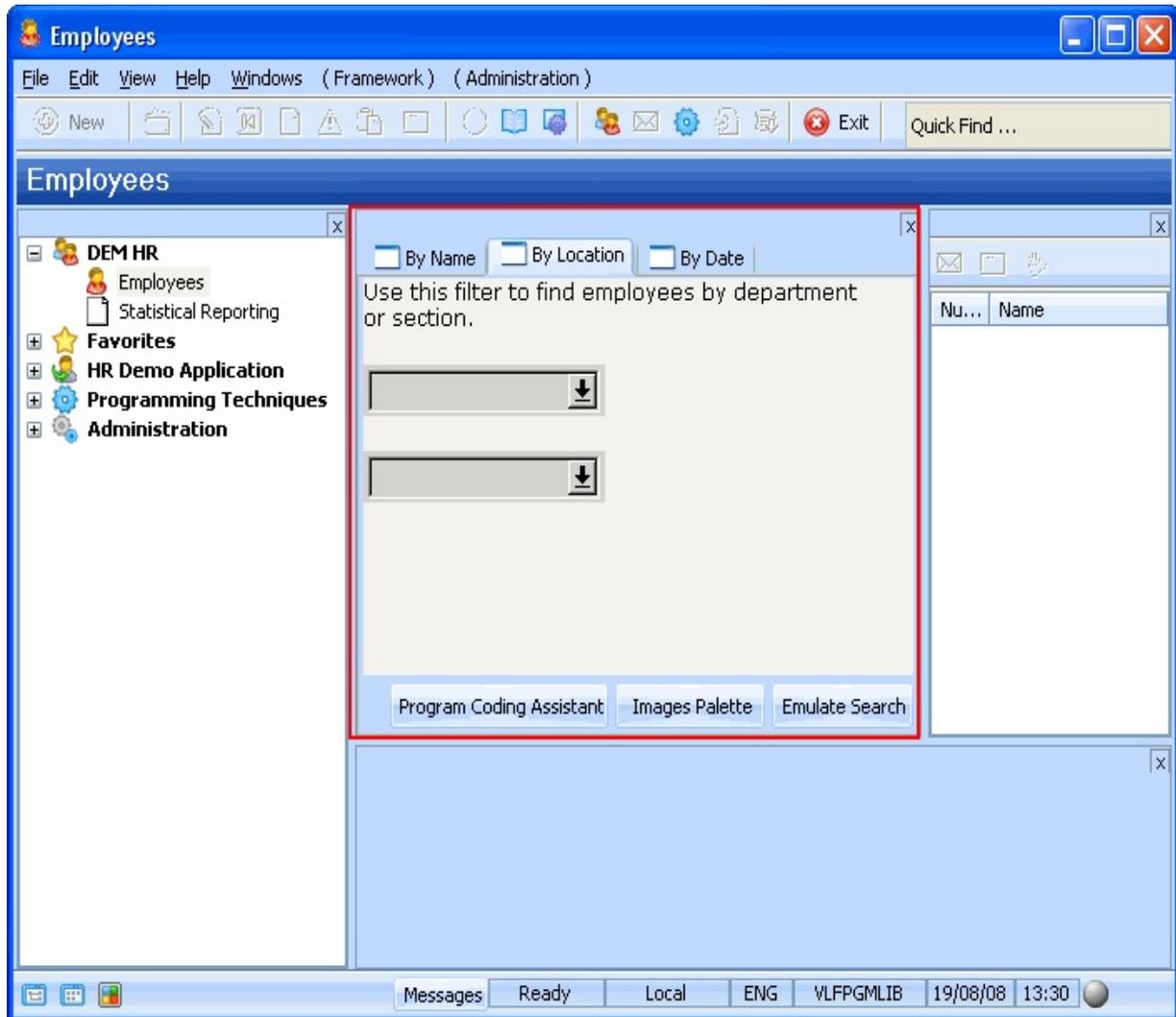
- The pad is actually a document containing lines. This means you cannot position pictures exactly. Use the Enter key to add new lines and add blanks or tabs to position items.
2. In the By Name filter delete the standard text and type in text indicating this filter lets the user select the employees by surname, first name or by employee number.
 3. To make your filter look more realistic, click on the Images Palette button to drag and drop or copy and paste some suitable pictures. Your finished prototype filter might appear something like this:



4. In the By Location filter, delete the standard text and type in text indicating

that this filter lets the user select the employees by department or section.

- Again, to make your filter look more realistic, you can click on the [Images Palette](#) button to drag and drop or copy and paste some suitable pictures. Your finished prototype filter might appear something like this:



- In the By Date filter, delete the standard text and type in text indicating that this filter lets the user select the employees by for example this criteria: Those who have their birthday today; Those that joined in the last 3 months; Those that have been with the company for more than 5 years.
- To make your filter look more realistic, you can click on the [Images Palette](#) button to drag and drop or copy and paste some suitable pictures.

Step 4. Filters for Statistical Reporting Business Object

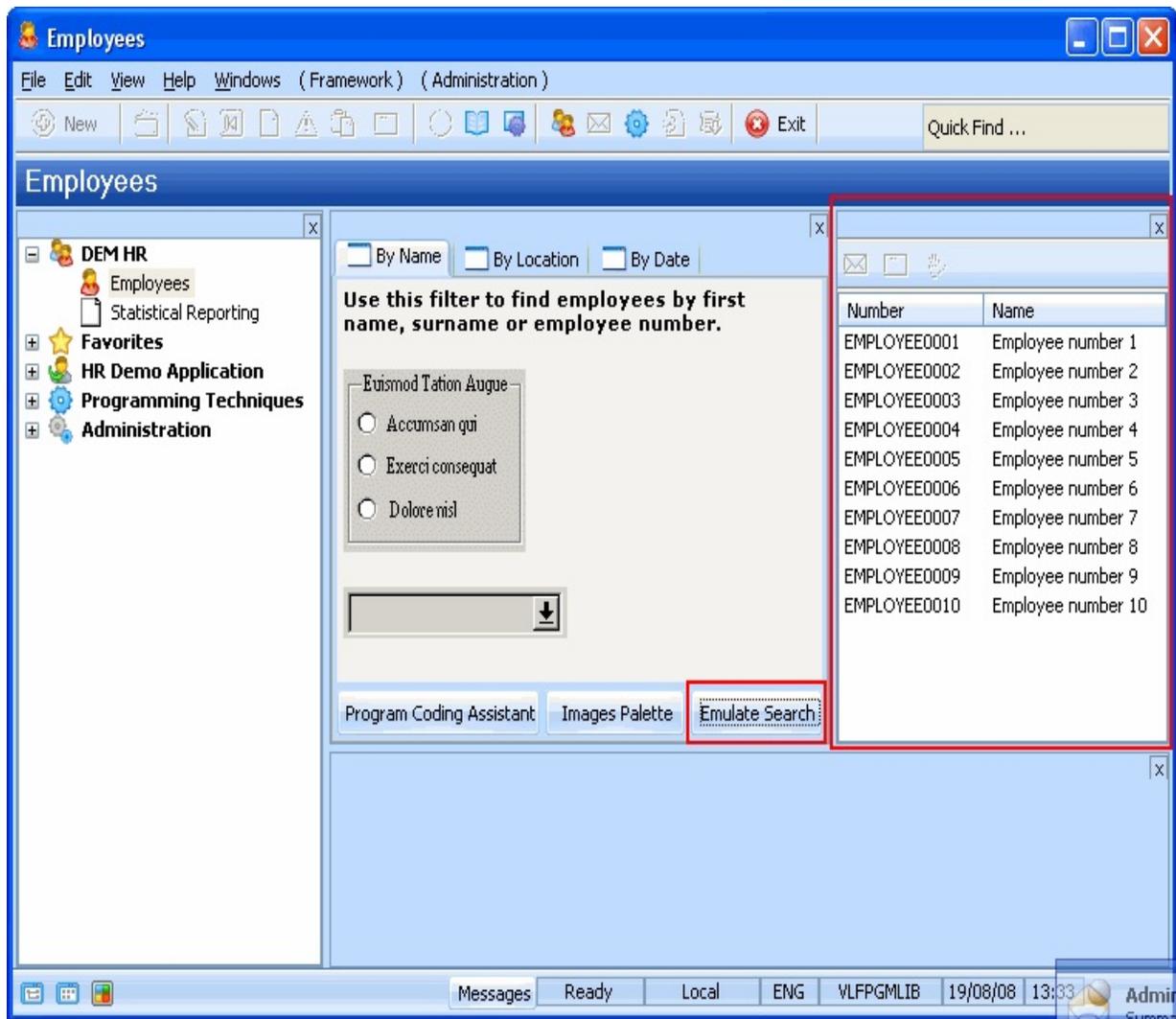
The Statistical Reporting business object does not need any filters. In [VLF004 - Prototyping Your Commands](#) you will see how this business object is implemented. It is important to understand that, even though filters and instance lists are very often the appropriate structure for working with end-user objects, they are not mandatory. For instance in the shipped demonstration application the Organizations business object does not use filters.

1. Delete the new filter which has been created for the Statistical Reporting business object by default. (Display the Properties for Statistical Reporting, bring up the Filters tab and delete the filter).

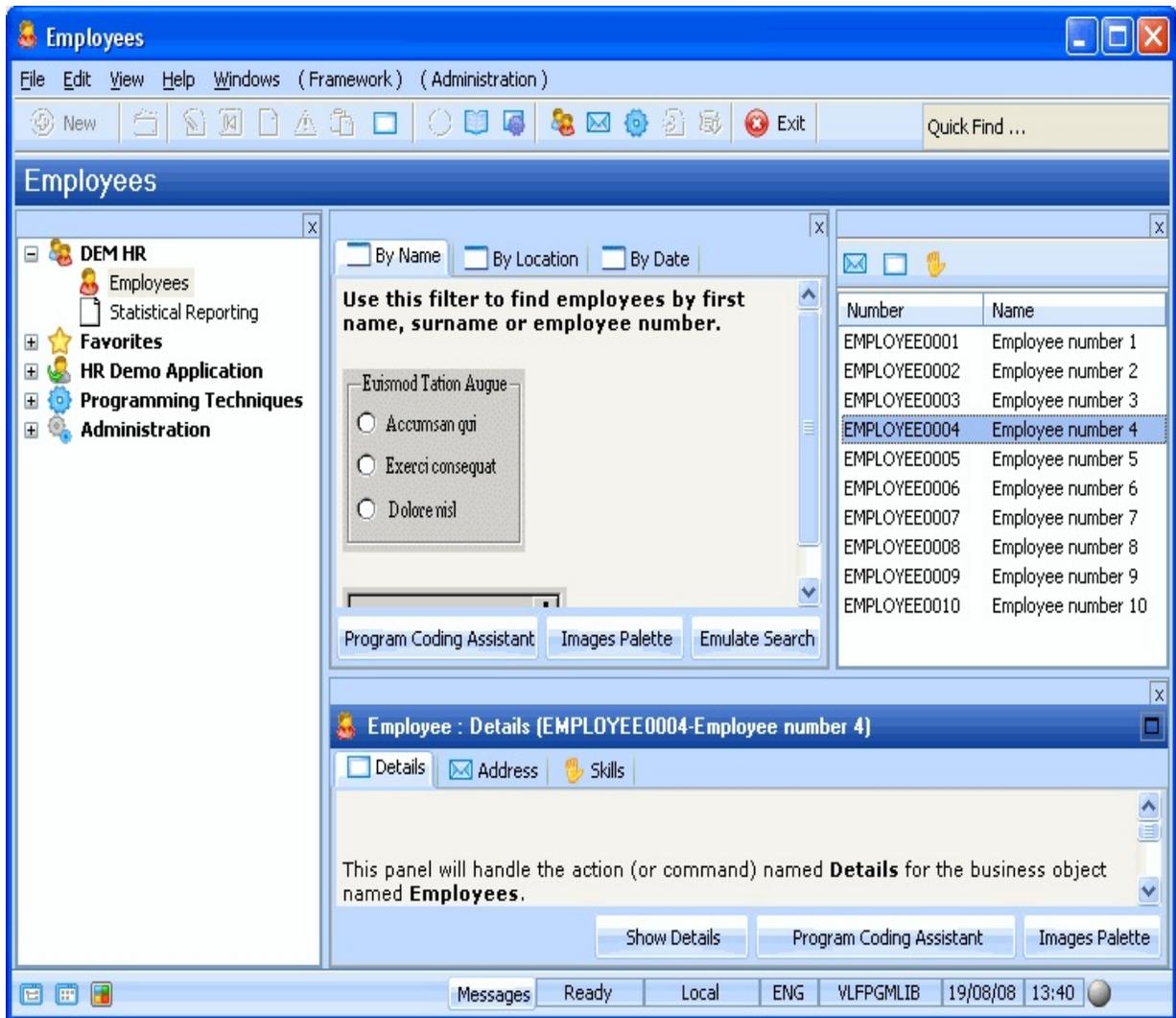
Step 5. View the Filters

You have now completed prototyping your filters. In a later tutorial, you will replace one of them with a real filter.

1. Now click the Emulate Search button on one of the filters. The items in the [Instance List](#) are updated (this is just a simulation, no filtering actually happens).



2. Click on an employee in the instance list to display the command handlers defined for it.



These are the commands you associated for the Employees business object when you created the prototype using the Instant Prototyping Assistant (see [Step 3. Specify Business Object Commands](#)).

Summary

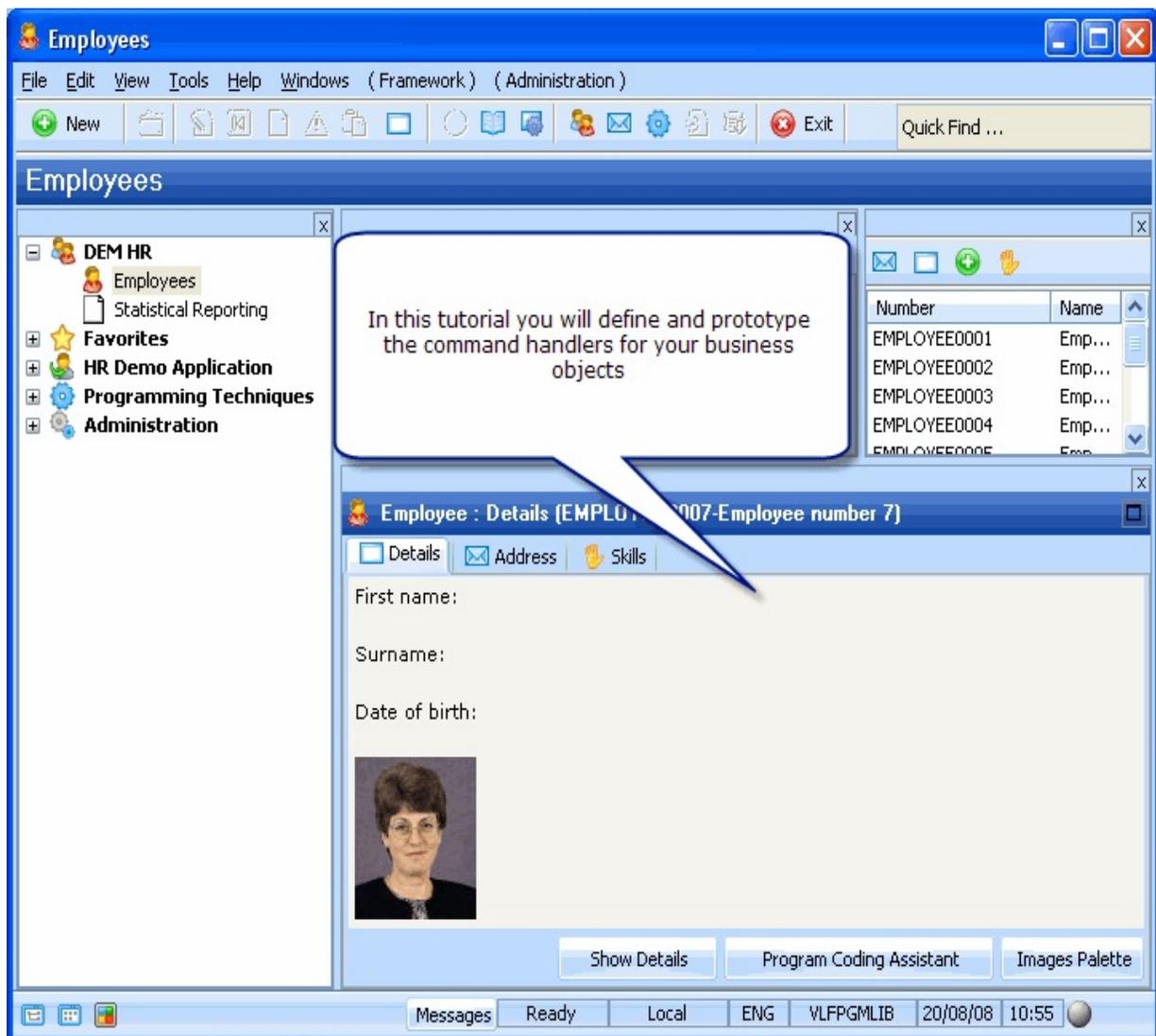
What I Should Know

- What filters do.
- How to define filters for a business object.
- How to prototype the filters by entering descriptive text and pictures using the [Images Palette](#).
- What the purpose of a filter is. You may want to review [Filters](#).
- What a mock up filter is. You may want to review [Mock Up Filters and Command Handlers](#).

VLF004 - Prototyping Your Commands

Objective

- To learn how to enable commands for a business object and business object instances.
- To learn how to prototype command handlers for the enabled commands.
- To add commands and command handlers for both the Employees and Statistical Reporting business objects.



To achieve these objectives, you will complete the following steps:

- [Step 1. View Command Definitions](#)
- [Step 2. Set Command Display for New](#)

- [Step 3. Prototype the New Command Handler](#)
- [Step 4. Prototype the Other Command Handlers](#)
- [Step 5. Define Commands for Statistical Reporting](#)
- [Step 6. Specify Command Display](#)
- [Step 7. Prototype Command Handlers for Statistical Reporting](#)
- [Summary](#)

Before You Begin

You may wish to review:

- [Command](#) in [Key Concepts](#).

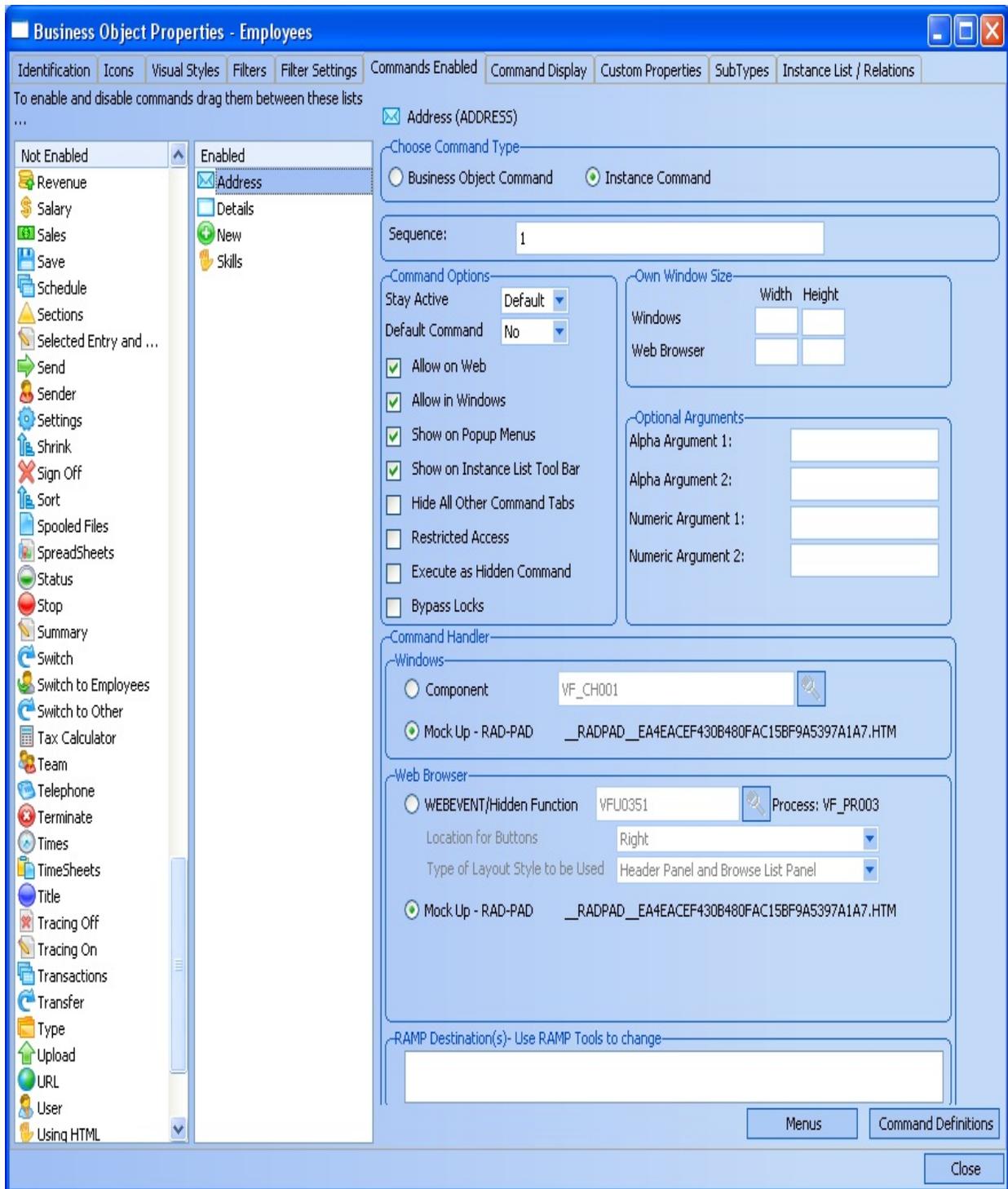
In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)

Step 1. View Command Definitions

In this step, you will view the definitions of the commands specified for the business objects.

1. Right-click the Employees business object and select Properties... option.
2. Display the Commands Enabled tab.
3. Examine the Commands Enabled tab. The command list on the right is the "Enabled" commands list. This is all the commands currently enabled for the Employees business object. The command list on the left shows the "Not Enabled" command list. This is all the commands within the Framework that are not currently enabled for the Employees business object.



To Enable a command drag it from the Not Enabled list into the Enabled list. To disable a command, drag it from the Enabled list into the Not Enabled list.

4. Select Details. Note that it is an **Instance Command** and a **Default Command**.

5. Select Address and Skills. Note that they are also instance commands.
6. Select New. Note that it is a [Business Object Command](#).

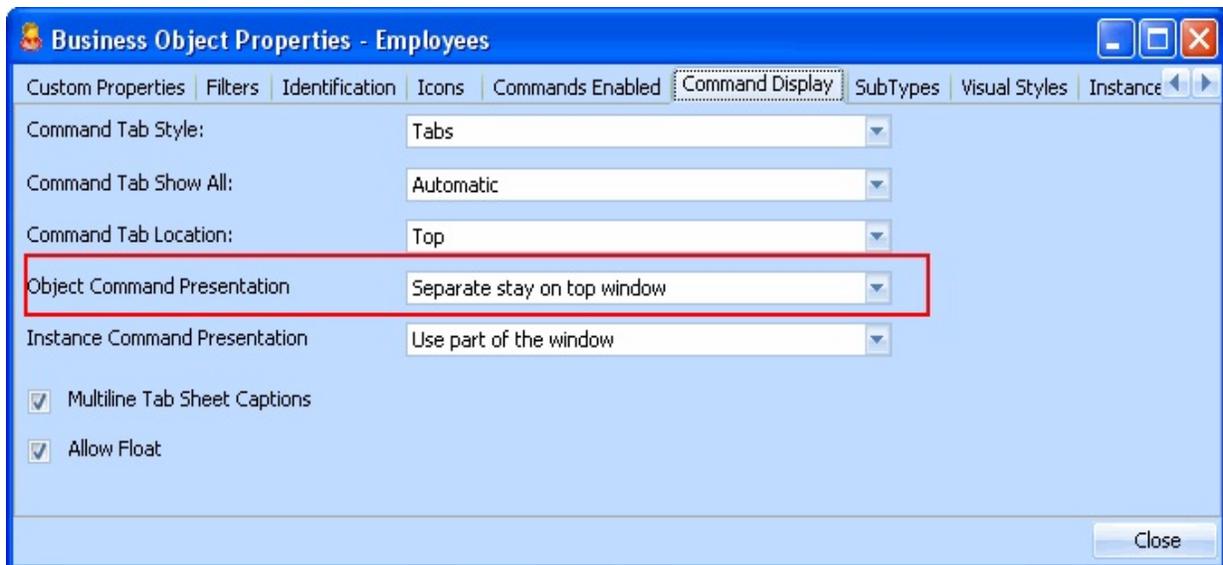
Make sure you understand the difference between these two types of commands. If you cannot find a command that you need, you can click on the Command Definitions button to add more commands. Note that when you click this button you are adding commands that can be used by any business object in the system. It is only when you assign a command to a specific business object that you define what the command does.

Step 2. Set Command Display for New

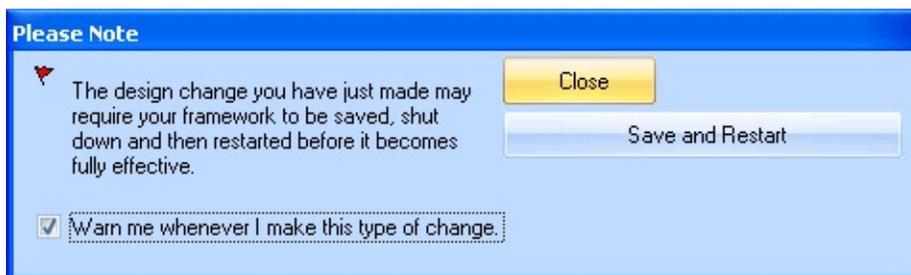
In this step, you will set the command handler for New to be displayed in a separate window when the user selects it.

1. Bring up the Command Display tab.
2. In the **Object Command Presentation** option, select Separate stay on top window.

Remember, New is the only command that is enabled for the business object Employees itself, the other commands are instance commands for Employees and will be displayed the command handler tab sheet.



When you change this option, the Framework prompts you to save your changes.

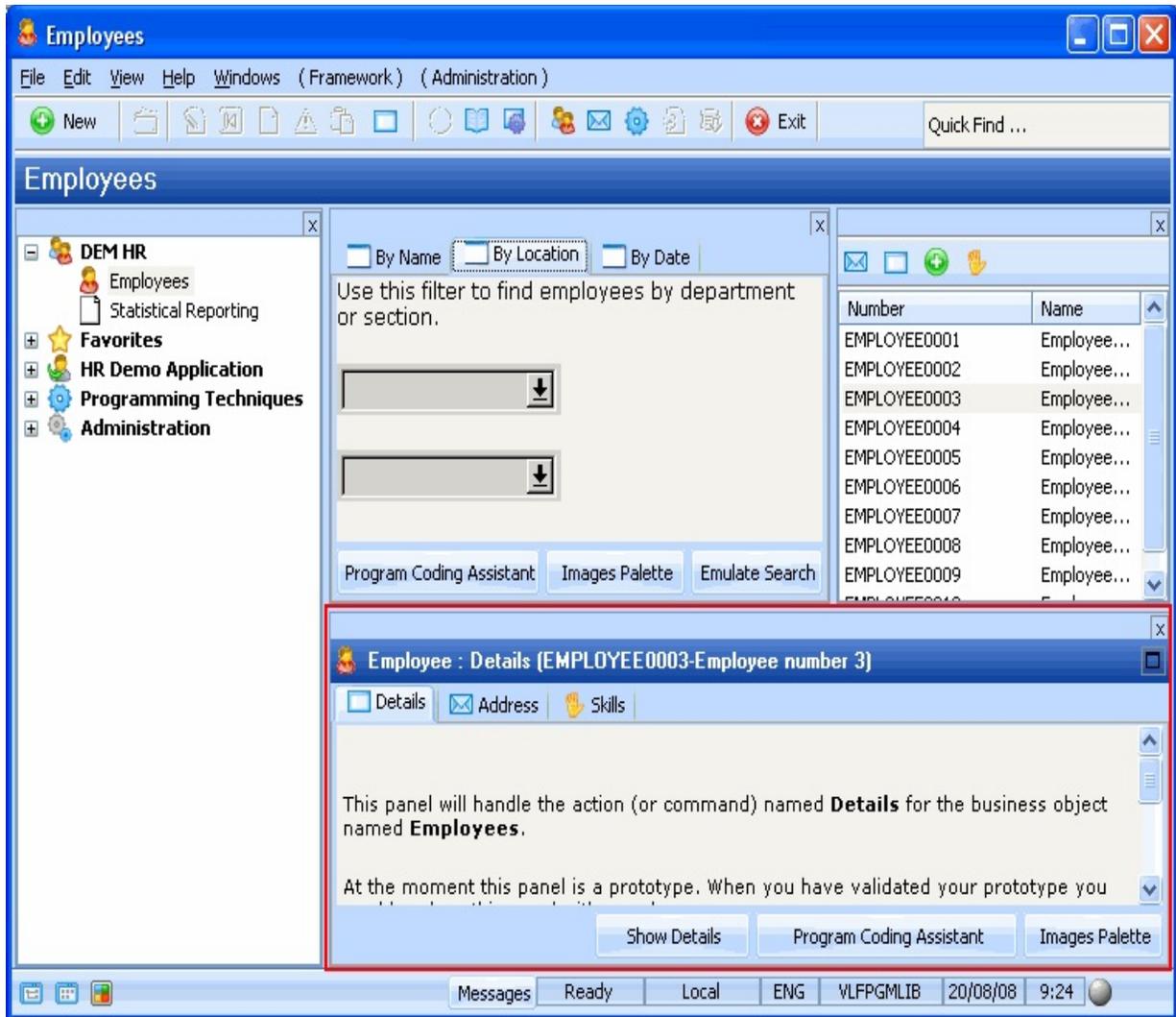


3. Click on the Save and Restart button.

4. Log on to the Framework.
5. To review your commands, select the Employees business object and right-click. The New command is visible in the pop-up menu.
If it is not visible, see the answer to the question [I have just changed my Framework design but the change has not taken effect?](#).
6. Select the New option from the menu. The command handler for New is shown in a separate window.



7. Close the New command handler window.
8. Click on the Emulate Search button in a filter and select an employee in the instance list. The tab sheets for Details, Skills and Address commands are shown.



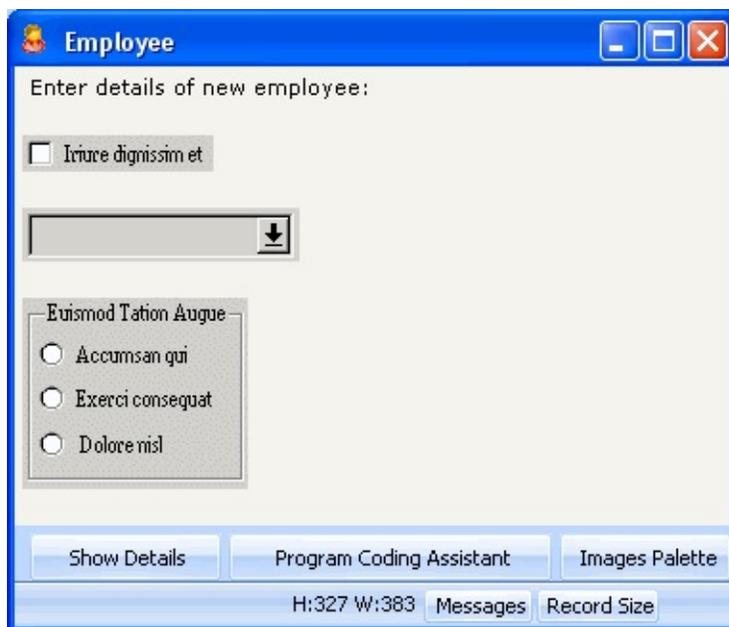
Step 3. Prototype the New Command Handler

To finalize your prototype, you need to model the handlers for these commands.

1. Select the Employees business object and then the New option from the File menu or the toolbar, or right-click Employees and choose New from the pop-up menu.

The [Mock Up RAD-PAD](#) command handler is displayed.

2. Select and delete the text from the prototype command handler.
3. Type in a note indicating that this command handler will request the user to enter mandatory details of a new employee.
4. If you want to, use the [Images Palette](#) to add pictures to the prototype command handler (do not attempt formal screen designs, just do a quick prototype):

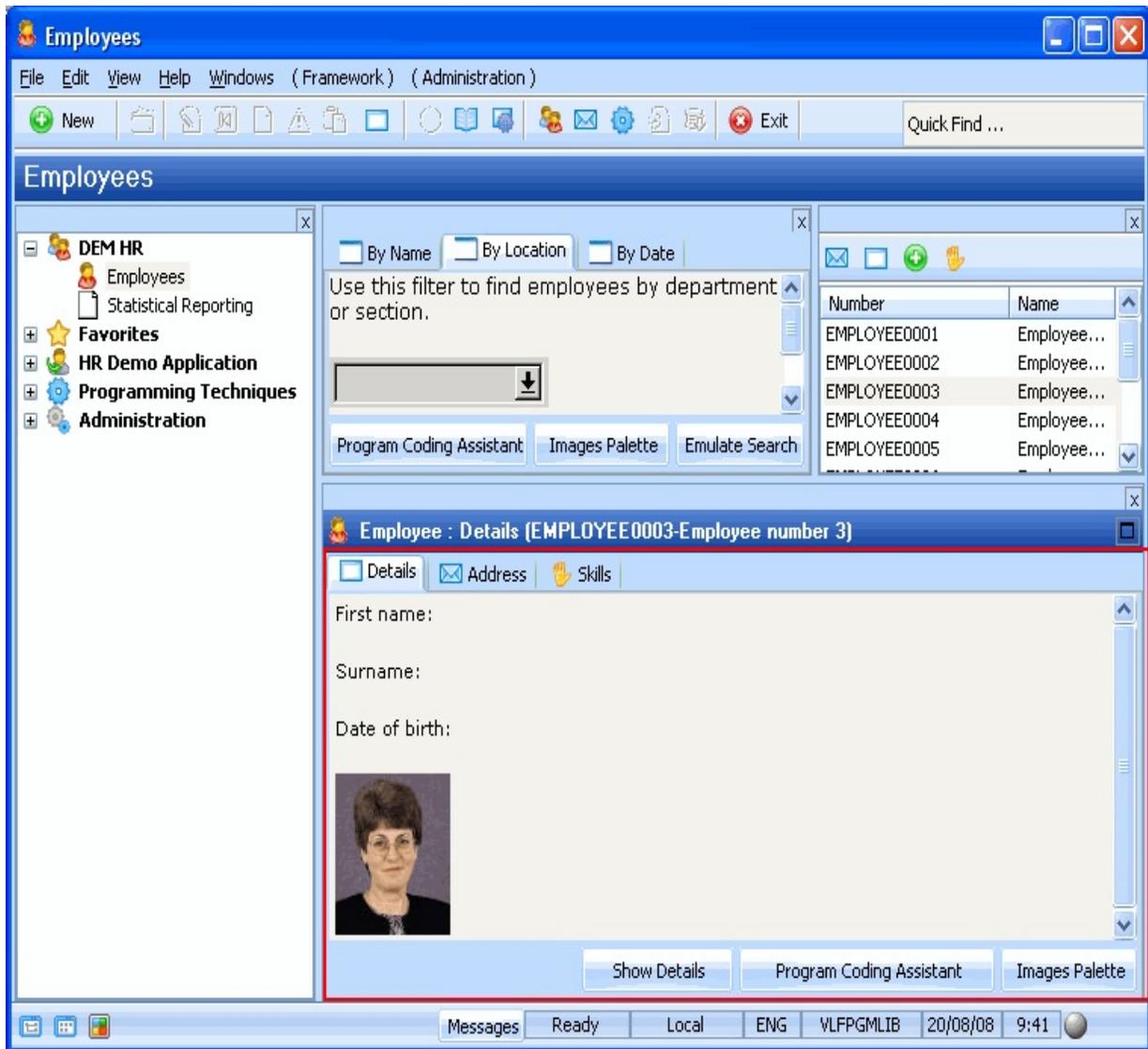


5. Close the command handler for New by clicking on the Close button in the top right-hand corner of the dialog box.

Step 4. Prototype the Other Command Handlers

In this step, you will prototype the handler for the Details, Skills and Address commands.

1. Select one item in the [Instance List](#).
2. A command tab folder showing the commands for the instances of Employees (Details, Address, Skills) is displayed.
Note that the tab for Details which is the [Default Command](#) for Employees is displayed topmost.
3. Design the prototype command handler for the Details command by typing in text and using the [Images Palette](#). Your prototype command handler for Details could look something like this:

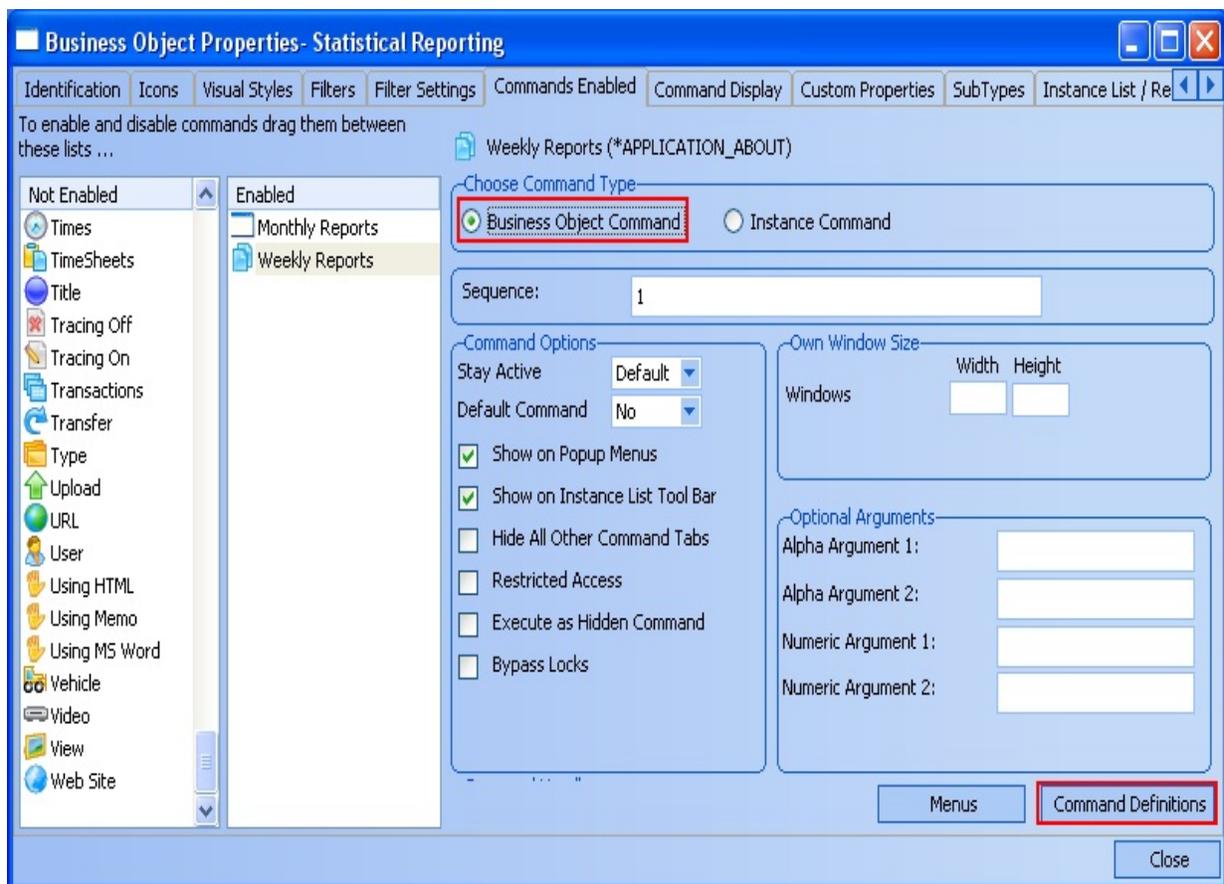


4. The Skills and Address command handlers also have Mock Up - RAD-PAD command handlers. Design them using the Images Palette and by typing in text.
5. You have now finished prototyping the command handlers for the Employees business object. Later on you will replace the prototype for Details with a real command handler.

Step 5. Define Commands for Statistical Reporting

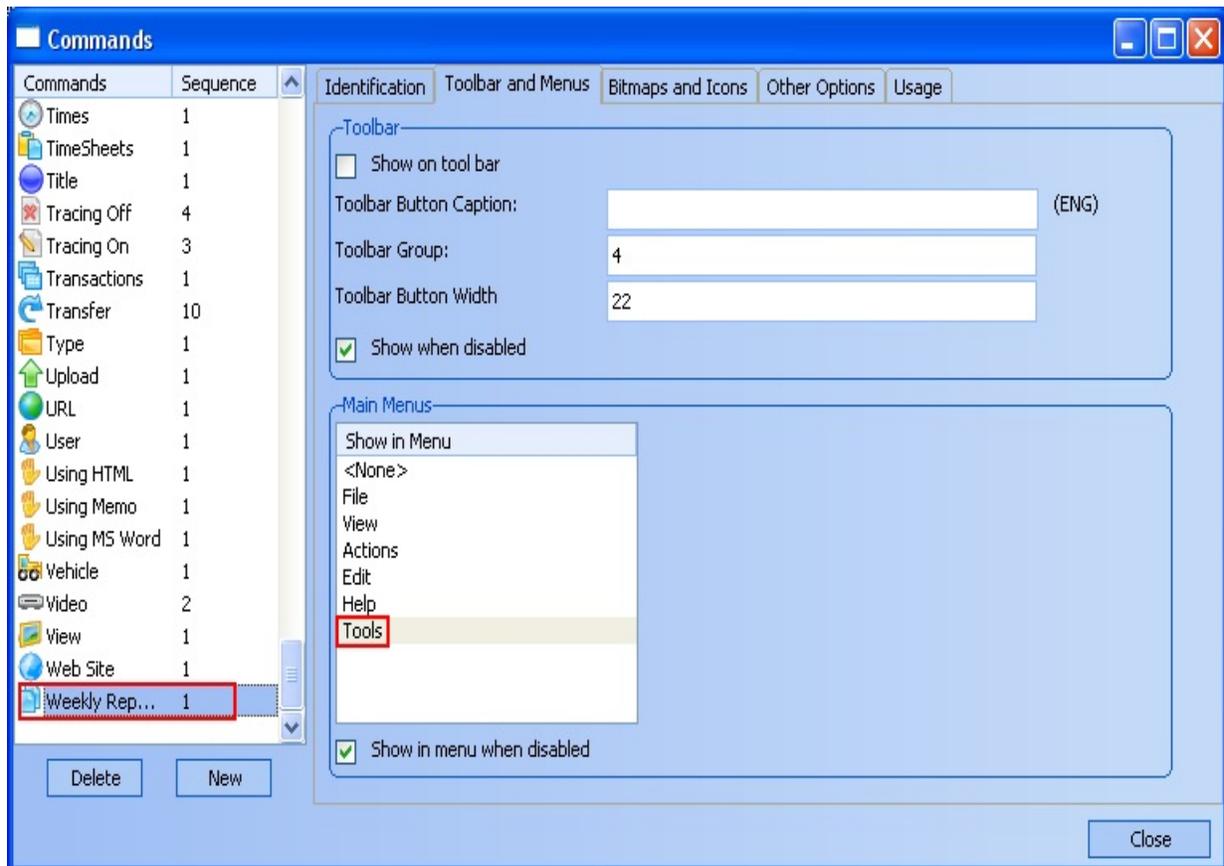
The Statistical Reporting business object will not have any filters or instance lists. Instead it will have two full-screen command handlers, Weekly Reports and Monthly Reports.

1. Double-click the Statistical Reporting business object to display its properties.
2. Click on the Commands Enabled tab. Notice that the two commands which you defined in [Step 3. Specify Business Object Commands](#), Weekly Reports and Monthly Reports, are enabled.
3. Make both commands Business Object commands. When a message appears prompting you to restart the Framework, just click Close.
4. Click on the Command Definitions button on the bottom right.



The Commands dialog box is displayed. This dialog box is used to set the high level definitions for each command defined in the Framework. These definitions are shared by all business objects that use the command.

5. Select the Weekly Reports command from the list on the left.
6. Bring up the Toolbar and Menus tab.
7. Add the Weekly Reports command to the Tools menu by selecting the command and then Tools in the [Show in Menu](#) list:

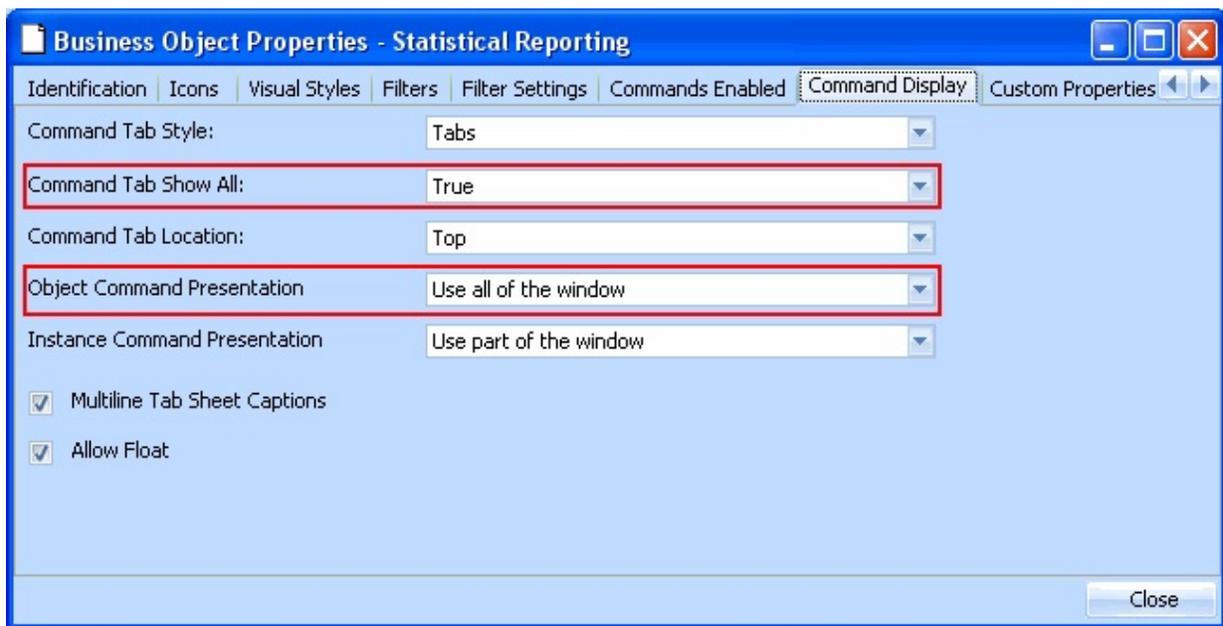


8. Repeat the above steps to add the Monthly Reports command to the Tools menu.
9. Click on the Usage tab to see which business objects use this command.
10. Select the Details command from the list on the left. Notice that it is used by many business objects, including the Employees business object.
11. Close the Commands dialog box.

Step 6. Specify Command Display

In this step, you will change the way the command handlers are displayed.

1. Bring up the Command Display tab.
2. Change the [Command Tab Show All](#) option to True so that the report handlers will be shown simultaneously.
3. Click the Close button when the Framework prompts you to save and restart.
4. Change [Object Command Presentation](#) to Use all of the window.



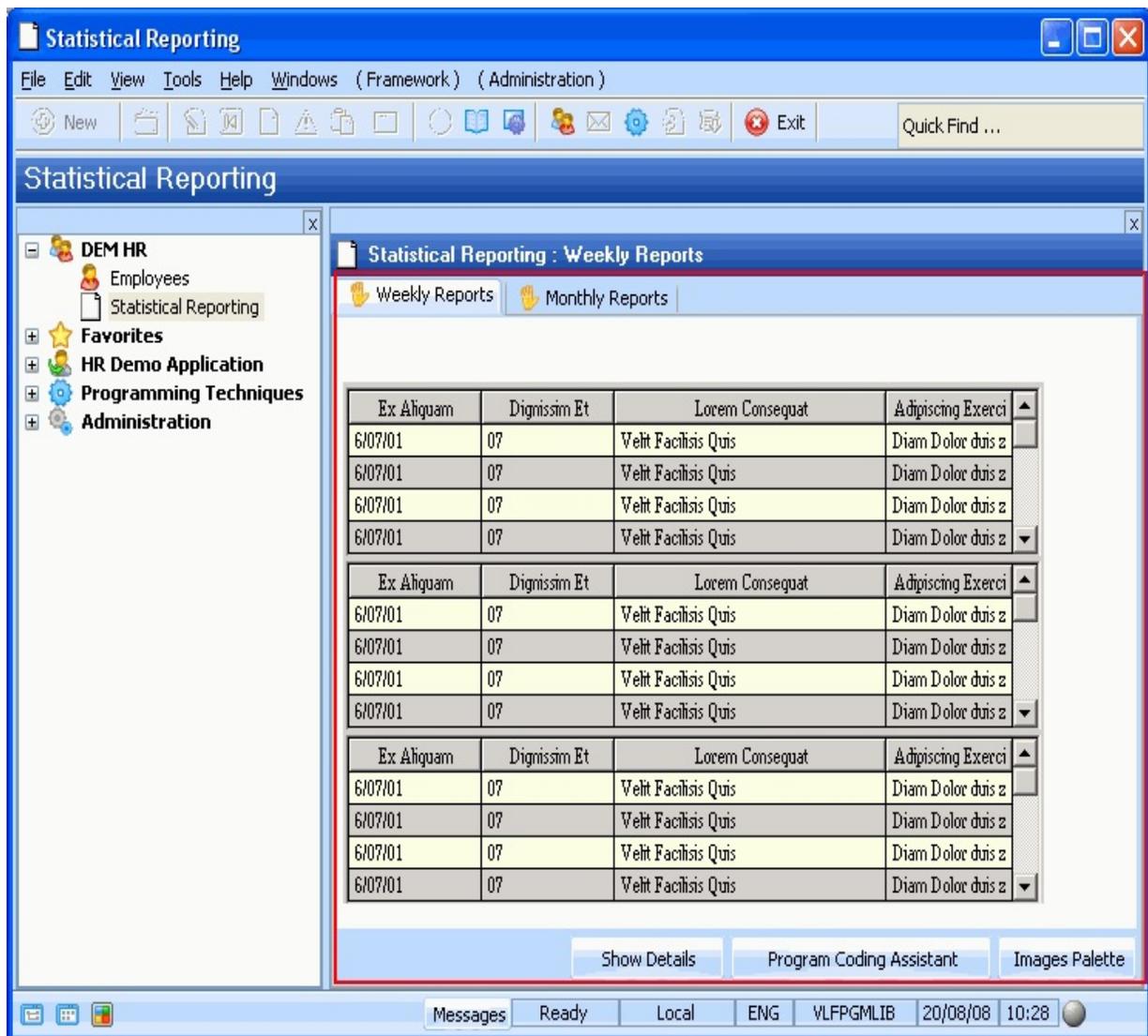
Note that the command display options apply to all commands associated with the Statistical Reporting business object.

5. Close the dialog box.
6. Close and restart the Framework.

Step 7. Prototype Command Handlers for Statistical Reporting

The following steps are optional, but if you prototype the command handlers, your application will look more complete. If you are comfortable with creating mock up command handlers, you can skip this step.

1. Select Statistical Reporting and then the Weekly Reports tab.
2. Select and delete the standard RAD-PAD text from the command handler prototype and type in text and paste in commands using the images palette. (You can also use other pictures, for example clip-art). Prototype the handler as you wish. The end result could look something like this:



You can prototype the other command handler in a similar way.

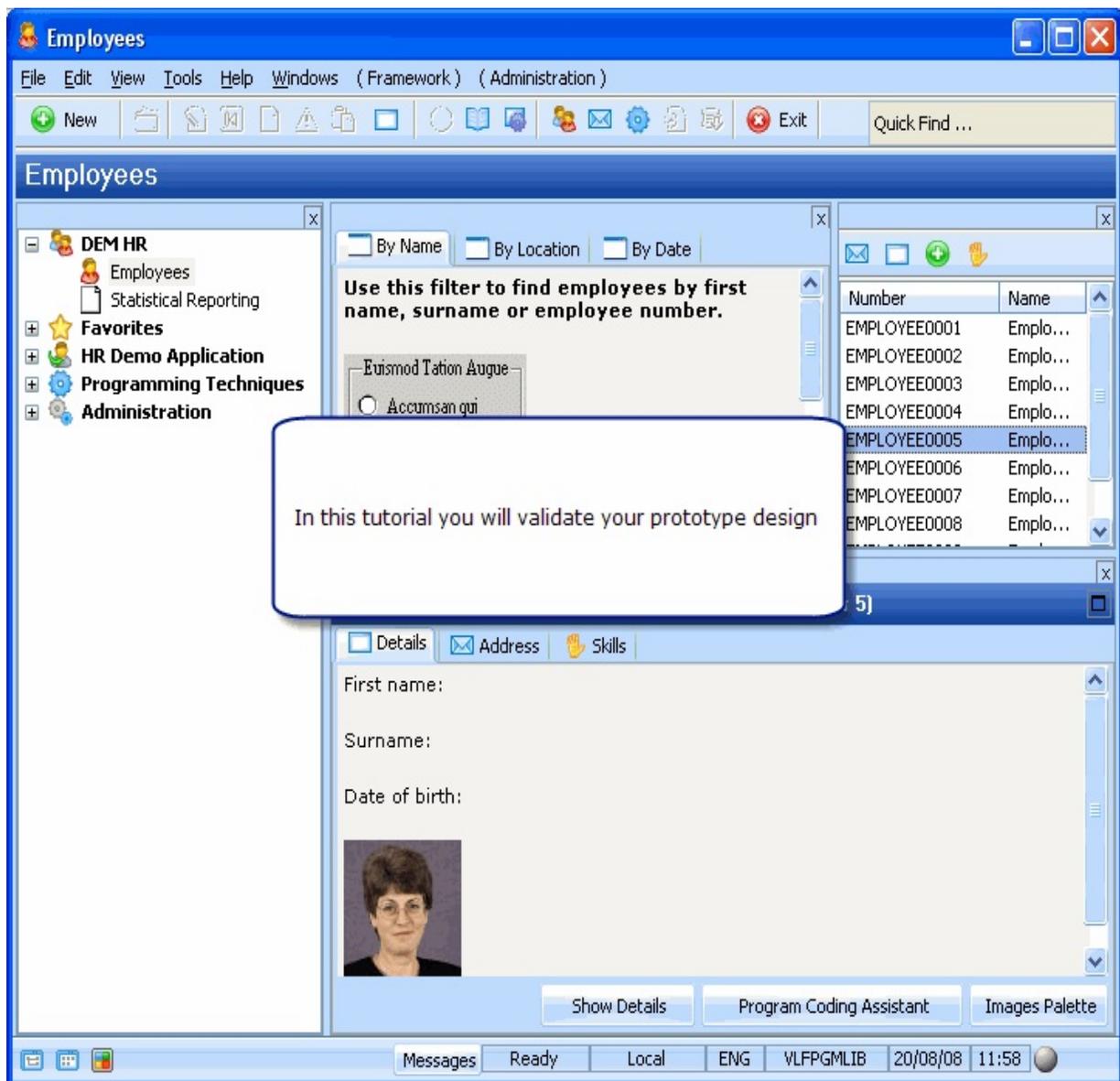
Summary

What I Should Know

- How commands are associated with business objects and business object instances.
- What a command handler is. You may want to review [Command Handler](#).
- How to prototype the command handlers by entering descriptive text and pictures using the Images Palette.
- What the difference between a business object command and a business object instance command is. You may want to review [Command](#) in [Key Concepts](#).
- What you need to do after you have created new commands.

VLF005 - Validating the Prototype Objective

- To remind you that, at this point, you should validate your prototype. Do not start writing any code before the basic structure of your application has been validated.
- To execute your application as a Visual LANSa application.
- Optional: To execute the prototype in Web mode.



To achieve this objective, you will complete the following steps:

- [Step 1. Validate Your Prototype in Windows mode](#) in Windows mode

- [Step 2. \(Optional\) Setting up your Environment for Web Browser applications](#)
- [Step 3. \(Optional\) Validate Your Prototype in Web mode](#)
- [Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)

Step 1. Validate Your Prototype in Windows mode

Your prototype is now finished. You will now want to see the prototype as the end-user will see it.

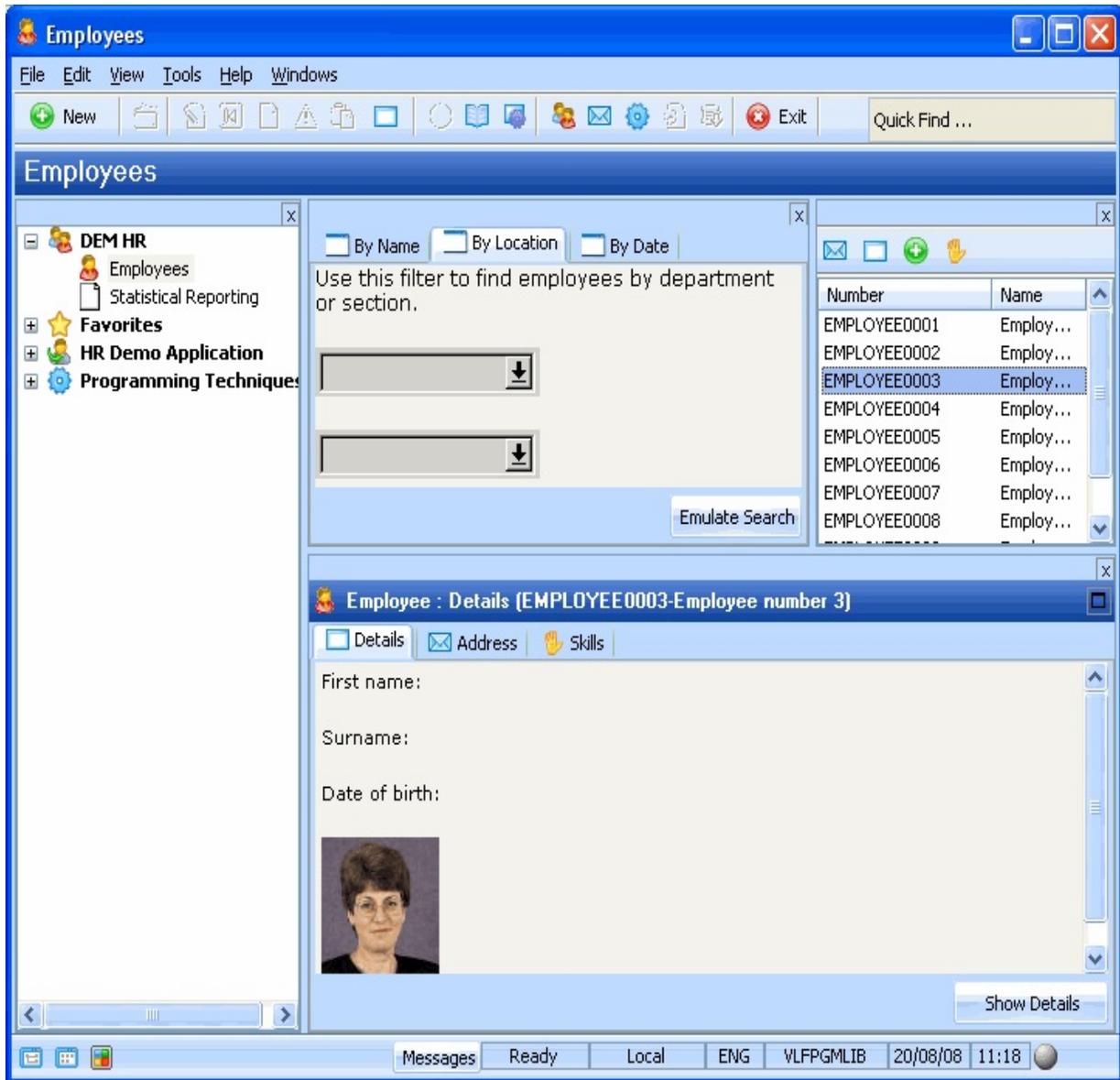
1. If you have not already done so, shut down the Framework. Be sure to save your changes.

2. Start the Framework as an end-user.

When you execute the Framework in end-user mode, the Framework and Administration menus are not displayed, nor are any design-time menu options in the popup menus (see [Menu Options in Brackets](#)).

3. Review your prototype. Notice that you do not need to compile anything to use your prototype.

- a. Select the III HR Application and view its business objects and commands.



- b. Select the Statistical Reporting business object and review its command handlers.
 - c. Select the Employees business object and review its command, filters, instance list and command handlers
4. If you had a list of end-user tasks available, you should now make sure that you have adequately addressed all the requirements.
 5. If you were prototyping a real application, now would be the time to let the end-users try out the prototype. Users typically find it easy to give their input

when they have a concrete sample of the system available.

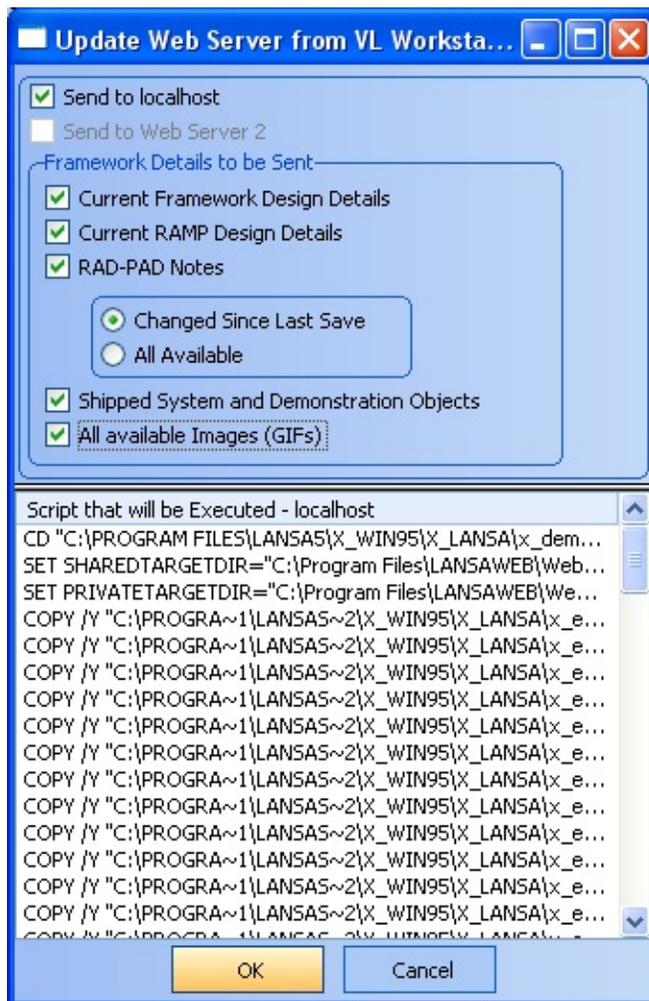
Step 2. (Optional) Setting up your Environment for Web Browser applications

If you want to see how your prototype would look when running on the web you need to configure your system.

1. Read [Setting Up Your Framework Environment](#) and follow the steps to set up your PC for web development.

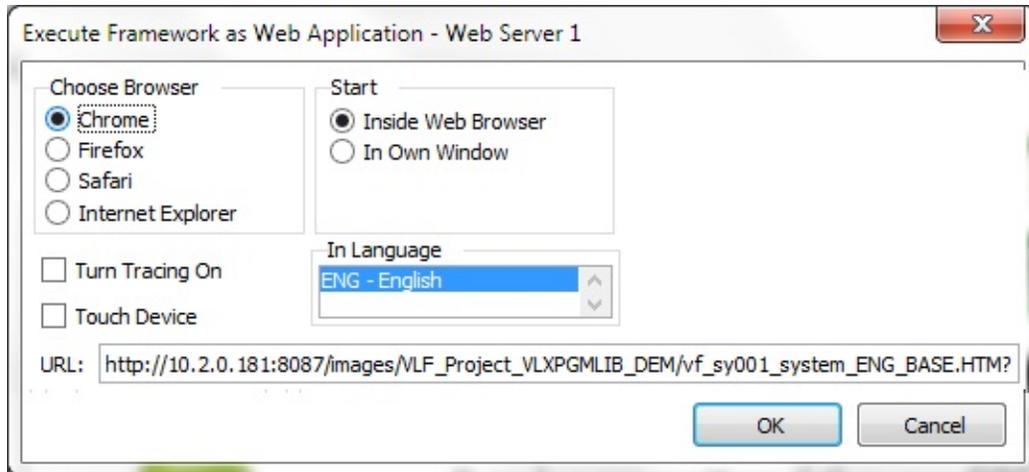
Step 3. (Optional) Validate Your Prototype in Web mode

1. Start by uploading your prototype to the web server:
 - a. Start the Framework as a designer.
 - b. Make a small change to the design (for example add or modify the hint of a business object)
 - c. In the (Framework) menu, select the (Save) option.
 - d. When the upload screen appears, ensure that it is set as shown below

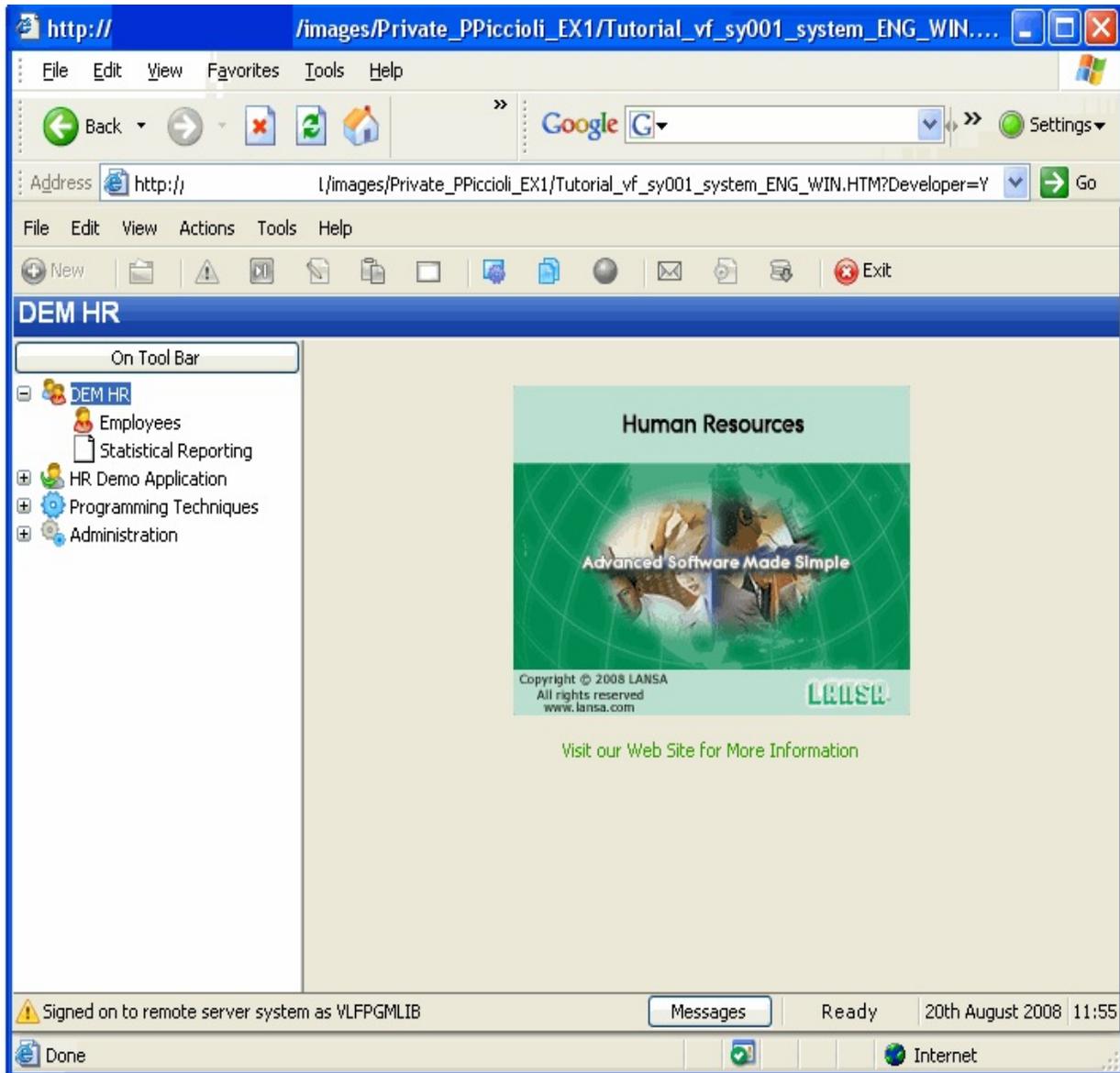


2. Wait for the upload to finish (It may take some time).
Check that all the upload messages are successful.
3. When it has finished, start the Framework in web mode as follows:

- a. In the (Framework) menu select the option to (Execute as Web Application)
- b. Accept the defaults and press OK.



- c. The prototype will appear in a browser window.



4. Review your prototype:
 - a. Select the III HR Application and view its business objects and command.
 - b. Select the Statistical Reporting business object and review its command handlers.
 - c. Select the Employees business object and review its command, filters, instance list and command handlers
5. If you had a list of user tasks available, you should now make sure that you

have adequately addressed all the requirements.

If you were prototyping a real application, now would be the time to let the end-users try out the prototype. All you have to do is send them the url from your browser, for example:

http://nnn.nnn.nnn.nnn/vlFramework/vf_sy001_system_BASE.HTM

and they can run your prototype from their desks.

Users typically find it easy to give their input when they have a concrete sample of the system available.

Summary

Important Observations

You have now completed prototyping your application. Using the prototype you can:

- Validate your design.
- Show it to end-users and others to obtain feedback.
- Quickly rework your design until it matches all the requirements.
- Create alternative solutions.
- Optionally run the prototype from the web server.

Note that at this stage the application is still a prototype. Actual functionality will be introduced with real filters and command handlers.



Tutorials for Windows Applications

Applies to **Windows** only.

Includes:

[VLF006WIN - Snapping in A Real Windows Filter](#)

[VLF007WIN - Snapping in A Real Windows Command Handler](#)

[VLF009WIN - Adding Instance List Columns in Windows Applications](#)

[VLF010WIN - Creating a Mini Filter](#)

[VLF011WIN - Creating a Parent Child Instance List](#)

[VLF012WIN - Controlling Navigation Using Switching and the Virtual Clipboard](#)

[VLF013WIN - Signaling Events](#)

[VLF014WIN - Debugging/Tracing](#)

After you have created and validated your prototype, you can develop it into a functional application. The basic structure and presentation of the application will remain unchanged as you continue to use the Framework. To complete the application, you simply replace the prototype filters and command handlers with real Windows ones.

In these tutorials, you will replace the employee filters with real filters and the Details prototype command handler with a real command handler:

Employees [Min] [Max] [Close]

File Edit View Tools Help Windows (Framework) (Administration)

New [Icons] Exit Quick Find ...

Employees

DEM HR

- Employees
- Statistical Reporting
- Favorites**
- HR Demo Application
- Programming Techniques
- Administration

By Name
 By Location
 By Date

Employee Surname:

Clear List

Number	Name
A0070	BROWN VERONICA
A0090	BLOGGS FRED JOHN ALAN
A1031	BLAKE JOHN
A3564	BROWN FREDDY

Employee : Details (A0070-BROWN VERONICA) [Min] [Max] [Close]

Details
 Address
 Skills

Employee Number	<input type="text" value="A0070"/>	Business Phone	<input type="text" value=""/>	<input type="button" value="Save"/>
Employee Surname	<input type="text" value="BROWN"/>	Start date (YYMM)	<input type="text" value=""/>	
Employee Given Name(s)	<input type="text" value="VERONICA ANN"/>	Termination Dat	<input type="text" value=""/>	
Street No and Name	<input type="text" value="12 Railway Street"/>	Department Coc	<input type="text" value=""/>	
Suburb or Town	<input type="text" value="Baulkham Hills"/>	Section Code	<input type="text" value=""/>	
State and Country	<input type="text" value="NSW Australia"/>	Employee Salary	<input type="text" value=""/>	
Post / Zip Code	<input type="text" value="2153"/>	Start Date (DDM)	<input type="text" value=""/>	
Home Phone Number	<input type="text" value="(02) 9609 4627"/>	Termination Dat	<input type="text" value=""/>	

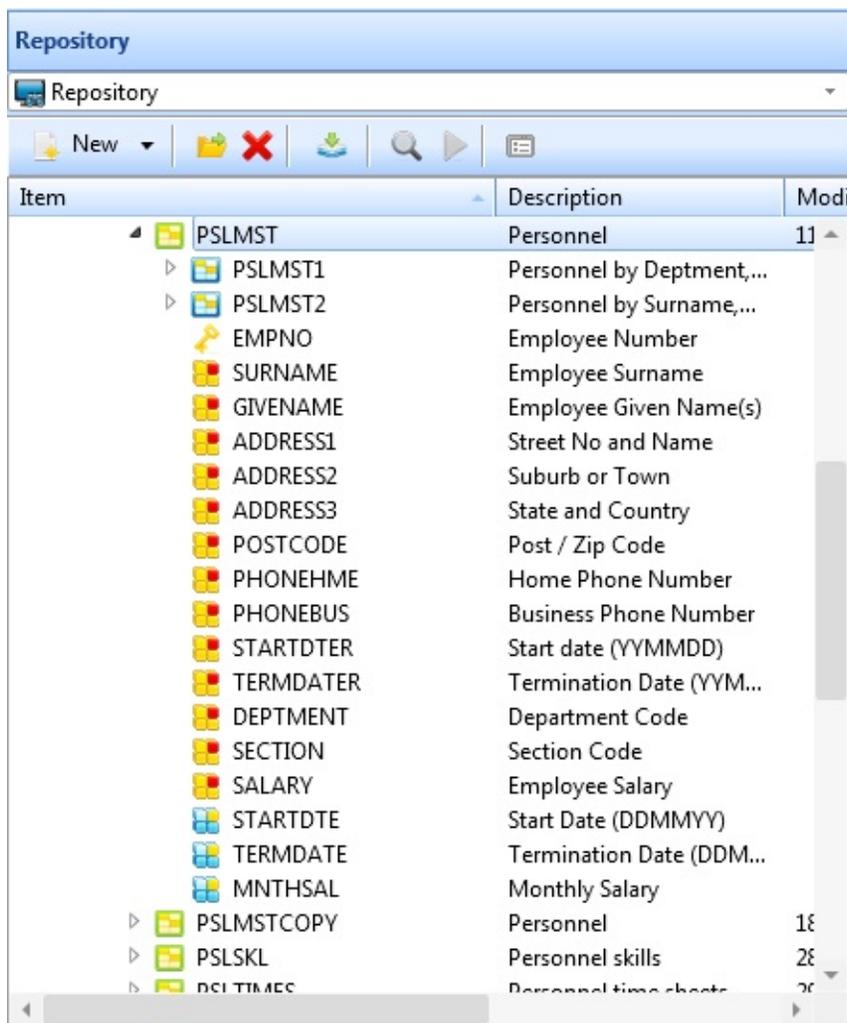
Messages Ready Local ENG VLFPGMLIB 22/08/08 13:46

The Personnel File

When prototyping your application, you decide your business objects based on an analysis of the tasks of the users of your application. At that point the database structure is not important.

Now that you are about to start implementing real filters and command handlers, you need to know how the data you will be using is stored.

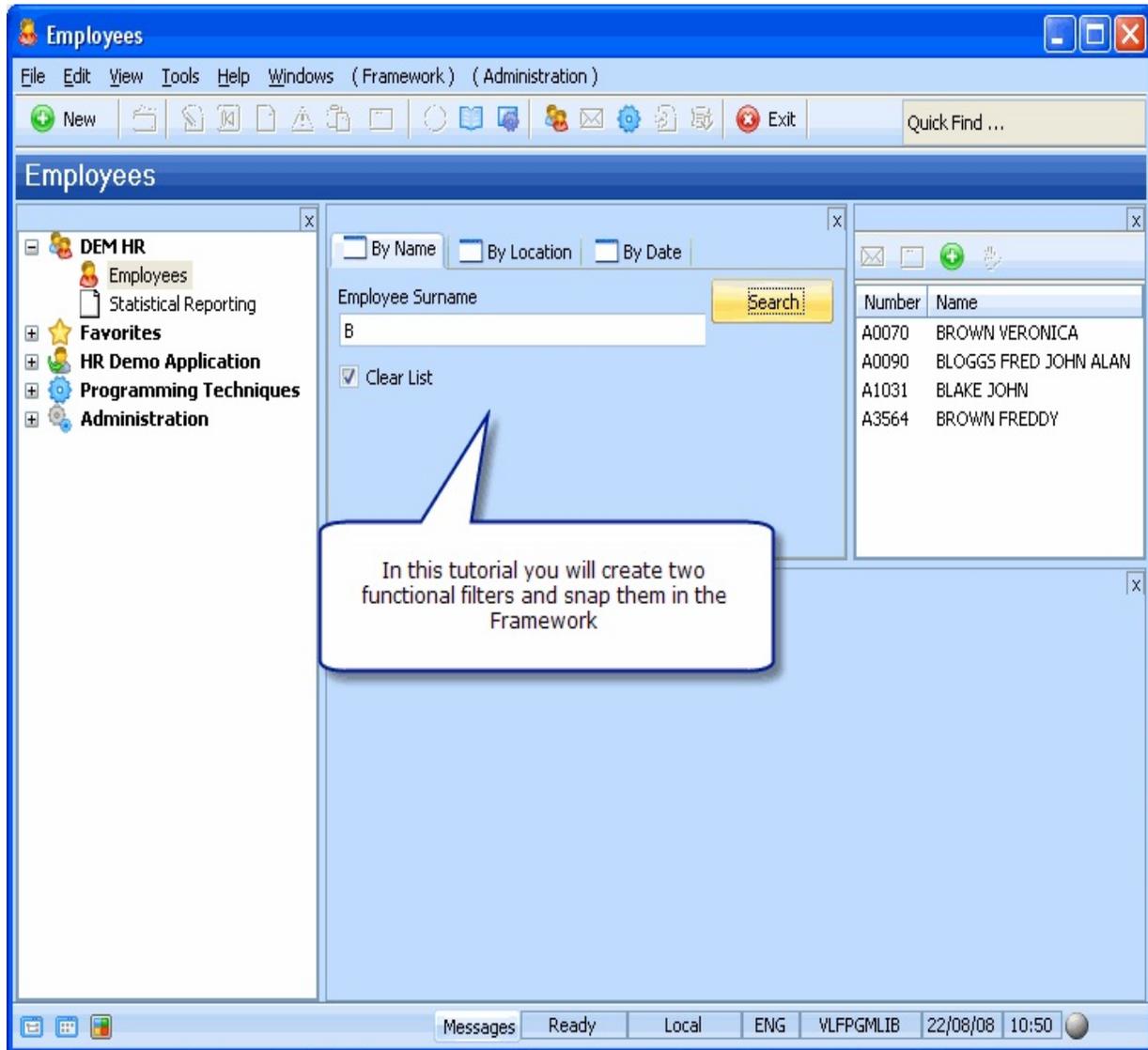
The following tutorials are based on the PSLMST Personnel demonstration file. Locate this file in the repository and view its contents:



Item	Description	Modi
PSLMST	Personnel	11
PSLMST1	Personnel by Department,...	
PSLMST2	Personnel by Surname,...	
EMPNO	Employee Number	
SURNAME	Employee Surname	
GIVENAME	Employee Given Name(s)	
ADDRESS1	Street No and Name	
ADDRESS2	Suburb or Town	
ADDRESS3	State and Country	
POSTCODE	Post / Zip Code	
PHONEHME	Home Phone Number	
PHONEBUS	Business Phone Number	
STARTDTER	Start date (YYMMDD)	
TERMDATER	Termination Date (YYM...	
DEPARTMENT	Department Code	
SECTION	Section Code	
SALARY	Employee Salary	
STARTDTE	Start Date (DDMMYY)	
TERMDATE	Termination Date (DDM...	
MNTHSAL	Monthly Salary	
PSLMSTCOPY	Personnel	18
PLSKL	Personnel skills	28
PSLTIMES	Personnel time sheets	30

VLF006WIN - Snapping in A Real Windows Filter Objective

- Learn how to replace prototype filters with real filters which will perform the actual selection of the items for the [Instance List](#).



To achieve this objective, you will complete the following steps:

- Step 1. [Creating Your Real By Name Filter](#)
- Step 2. [Snapping In the By Name Filter](#)
- Step 3. [Filter Code](#)
- Step 4. [Creating a Real By Location Filter](#)

- [Step 5. Snapping in the By Location Filter](#)
- [Summary](#)

Before You Begin

You may wish to review:

- [Filters](#) in [Key Concepts](#)
- [Framework Programming](#)

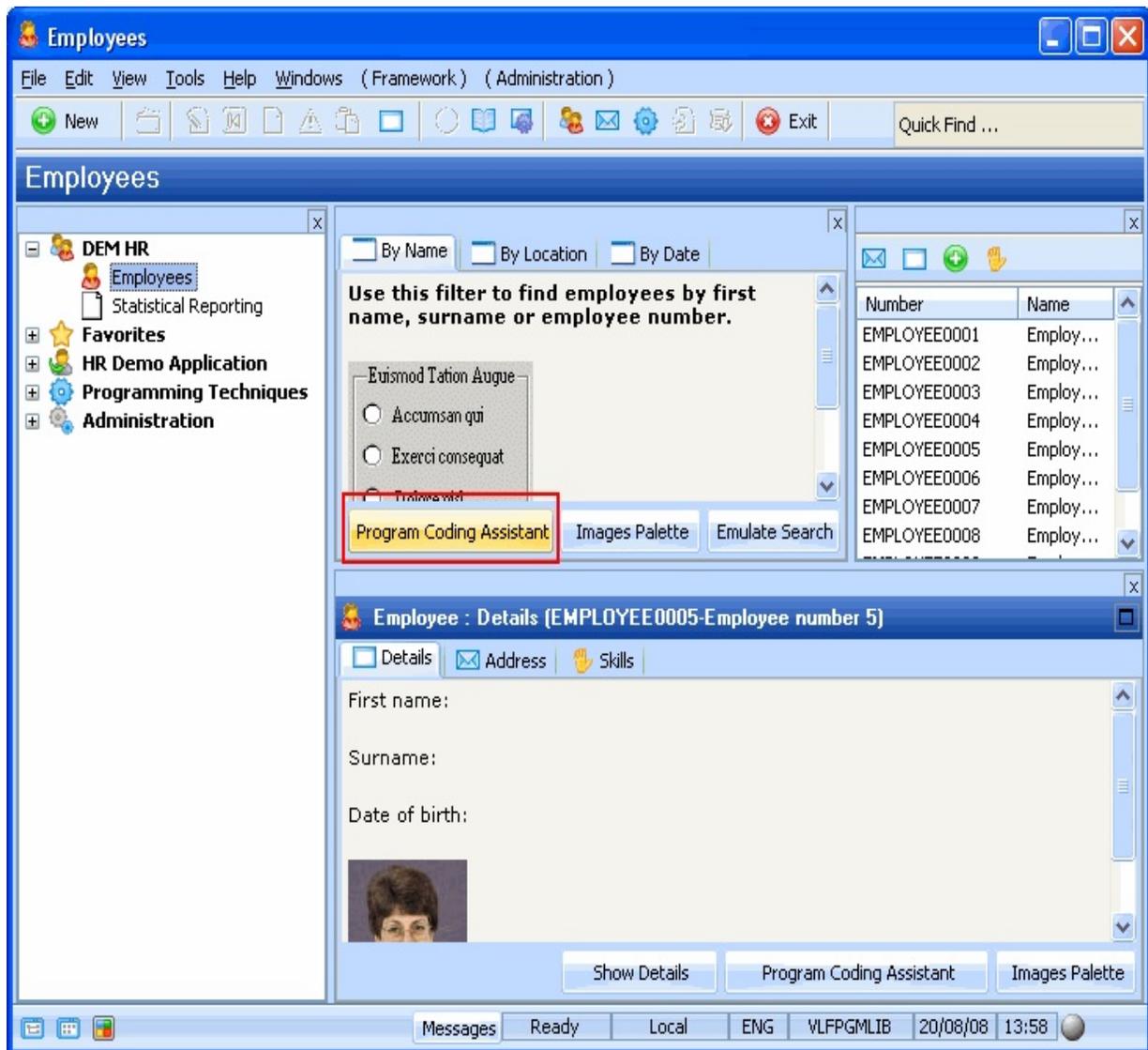
In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)

Step 1. Creating Your Real By Name Filter

In this step, you will create a real filter which searches the PSLMST file by employee surname. You will also learn how to use the [Program Coding Assistant](#).

1. Click on the Program Coding Assistant button in the By Name filter.



The Program Coding Assistant window is displayed. It allows you to create different types of components that can be plugged into your filters, instance lists and command handlers. It is highly recommended to use the program

coding assistant when you first start using the Framework.

Initially you will most likely use filters that generate a component that can be executed (e.g. CRUD Filter (Create/Read/Update/Delete), Filter that searches a file or view). As you progress you might only use a skeleton filter or simply copy from one that is similar to one that you want to create.

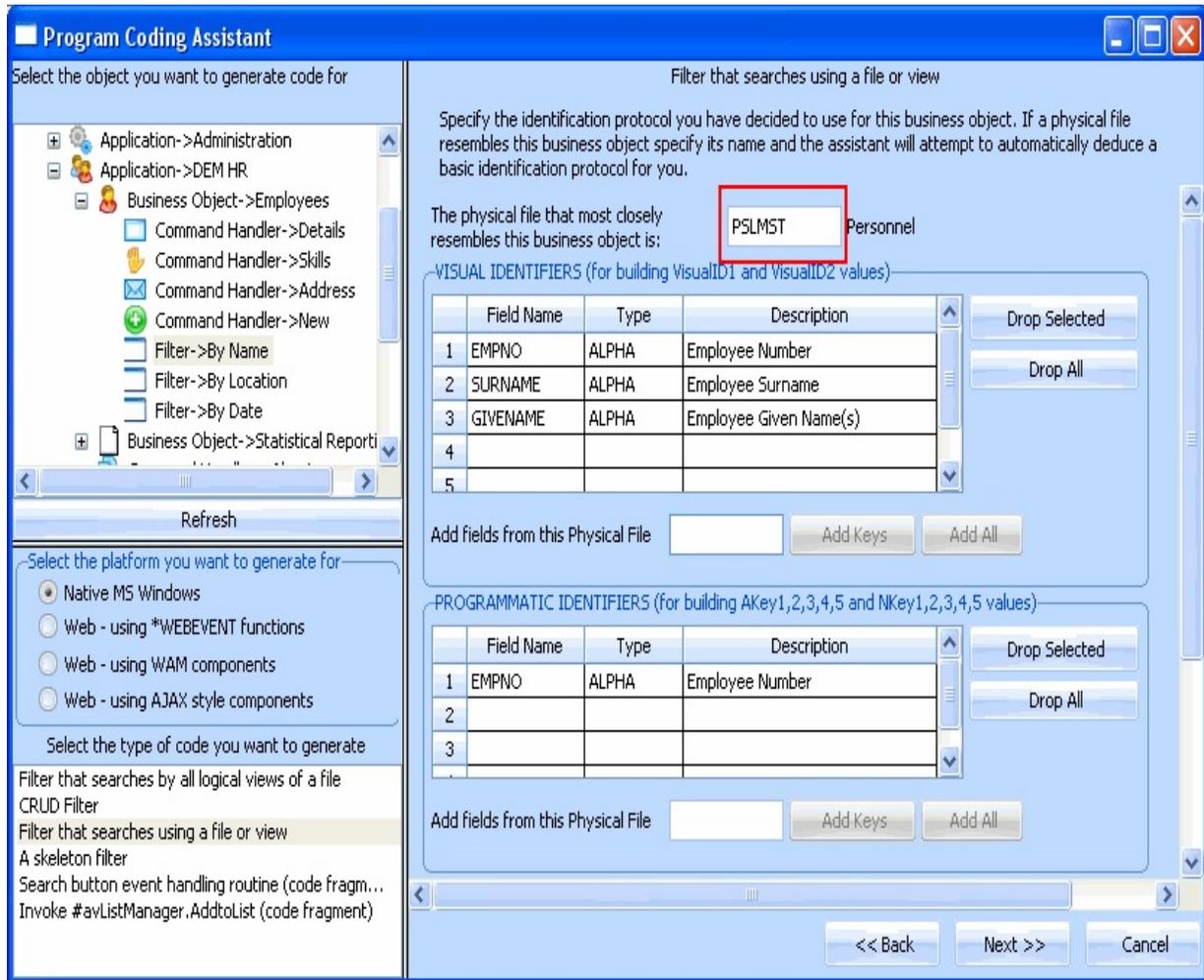
2. If you are using a non-English system, click on Framework -> Your Framework in the top-left tree view. The Set LANSA code generation preferences option appears at the bottom. Select this option and set your preferences.
3. In the list on the top left, select the iii HR application and then the By Name filter.
4. Underneath it, select Windows as the platform.
5. As the type of code you want to generate, select Filter that searches using a file or a view.

The screenshot shows the 'Program Coding Assistant' window. On the left, a tree view shows the selection path: Application->DEM HR -> Business Object->Employees -> Filter->By Name. Below this, 'Native MS Windows' is selected as the platform, and 'Filter that searches using a file or view' is selected as the code type. The main area displays a 'What?' section explaining that the assistant produces code for a filter that searches for information using a specified physical file or logical view. It notes that filters are used to dynamically create business object instance lists and provides an example of a Visual LANSA Framework filter window. This example window shows a table of employees with columns for Number and Name, and a search field for Employee Surname. The search field is circled in red. The table contains the following data:

Number	Name
A0070	VERONICA BROWN
A0090	FRED JOHN ALAN BLODGES
A1001	JOHN BLAKE
A2564	FREDDY BROWN

At the bottom of the window, 'Next >>' and 'Cancel' buttons are visible.

6. Click the Next button.
7. On the next page specify PSLMST as The physical file that most closely resembles this business object.



The Program Coding Assistant detects the Visual and Programmatic Identifiers required:

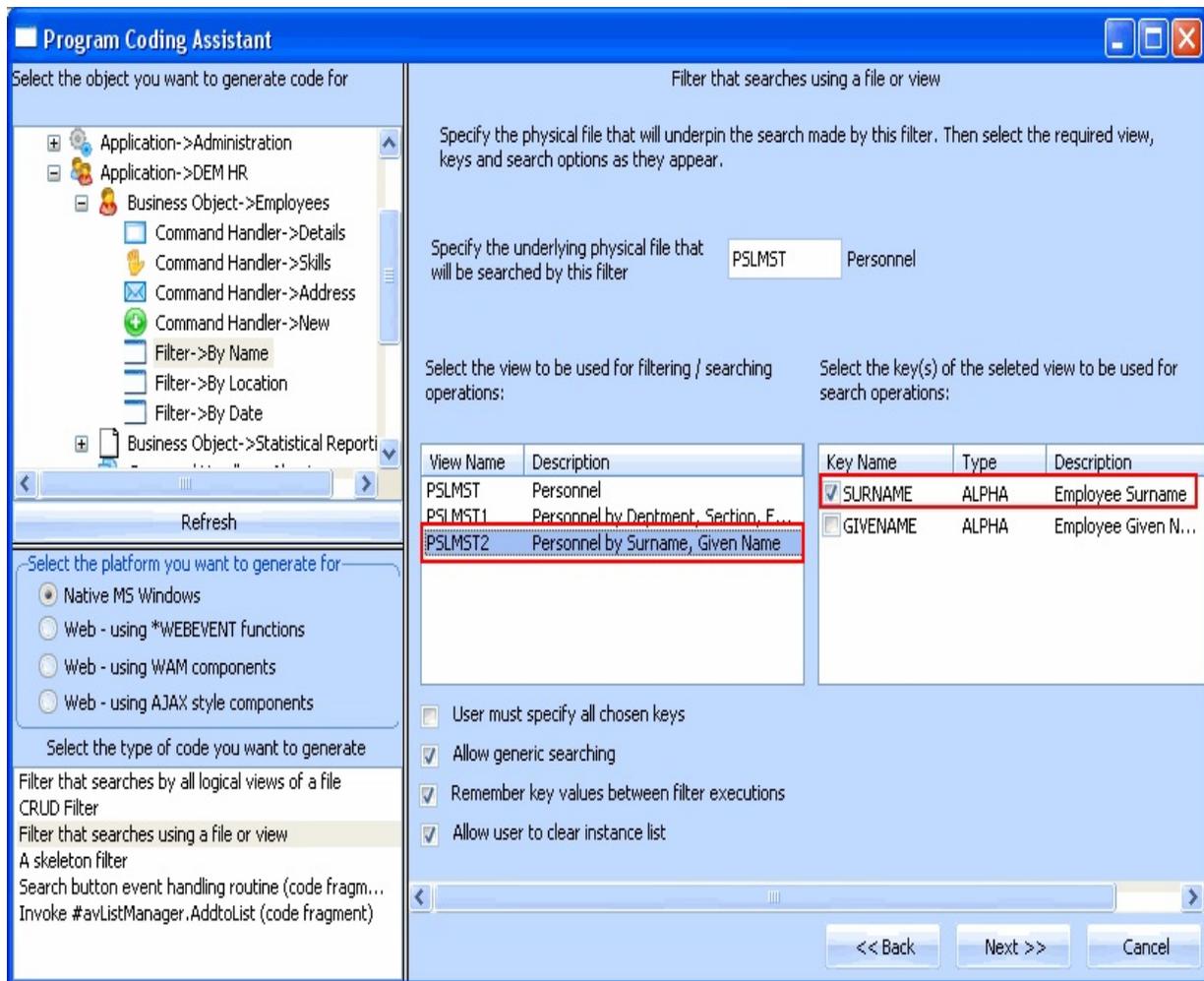
- A Visual Identifier is the field or fields that a user would use to identify a unique instance of the business object.
- A Programmatic Identifier is the field(s) that the program would use to identify a unique instance of the business object. Typically these would be the primary keys of the file or files that make up the data in the instance list.
- The additional columns represent the additional columns in your instance list

that you may have added during the prototyping phase.

8. Click the Next button.
9. On the next page specify PSLMST2 as the view to be used for filtering/searching operations. It is logical view of the PSLMST file keyed by the SURNAME and GIVENAME fields.

Note that you need an appropriate logical file for each filter that you want to create. Before implementing all your filters, review your data model to confirm that all the logical files exist. Doing so will speed up the process of implementing your prototype.

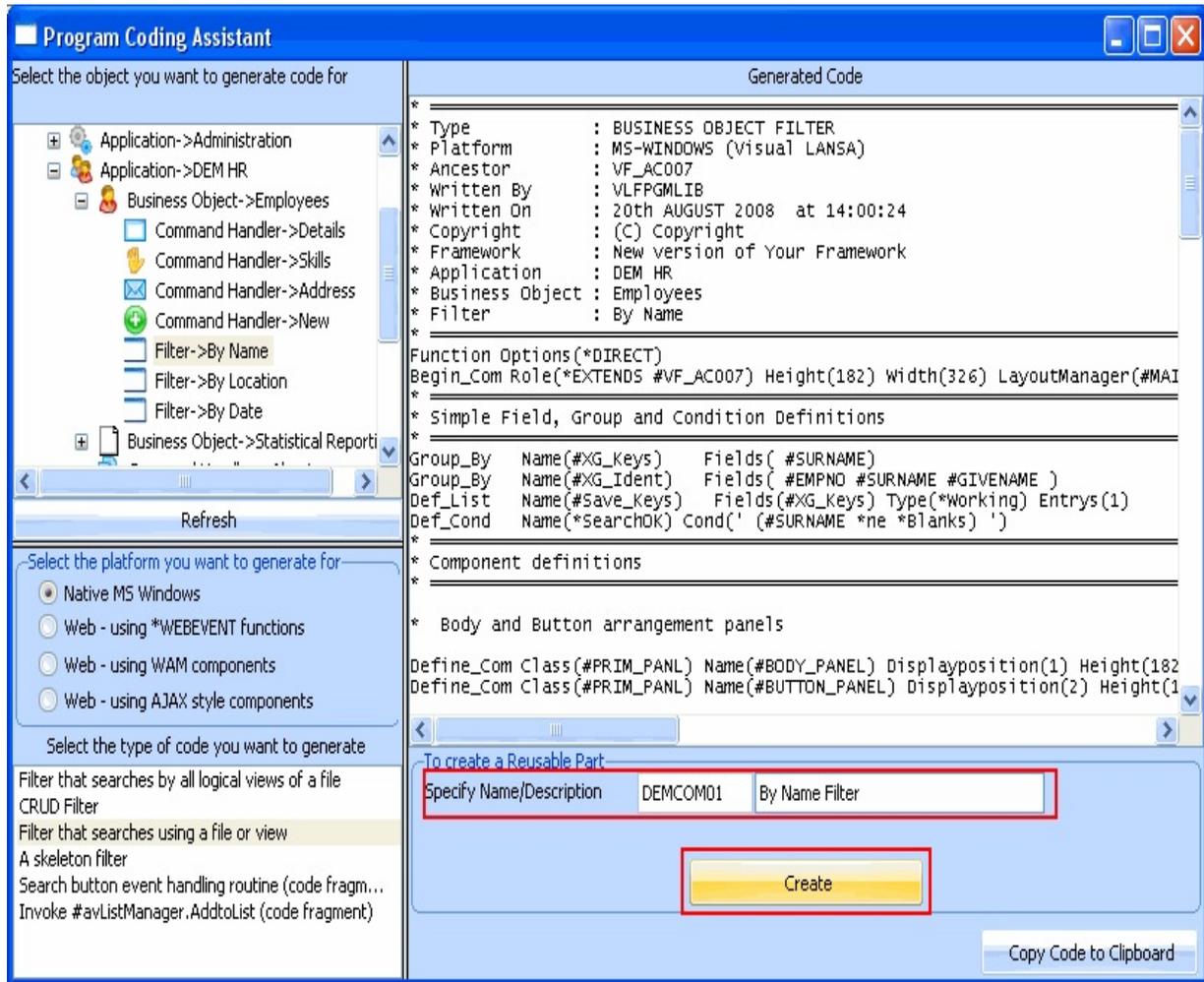
10. Select the SURNAME field as the key of the view to be used for search operations.



11. Click the Next button. Ignore the options on this page.
12. Click the Next button.
13. On the next page click the Generate Code button.

The next page, Generated Code, displays the source code for your filter. You now need to create the component that will contain this code:

14. Specify iiiCOM01 as the name of your real filter and By Name Filter as the description. (iii are your initials If you are using an unlicensed or trial version of Visual LANSA, you must always use the 3 characters DEM to replace iii).
15. Click on the Create button to create the component.



After a brief delay the Filter component is displayed in the Visual LANSA

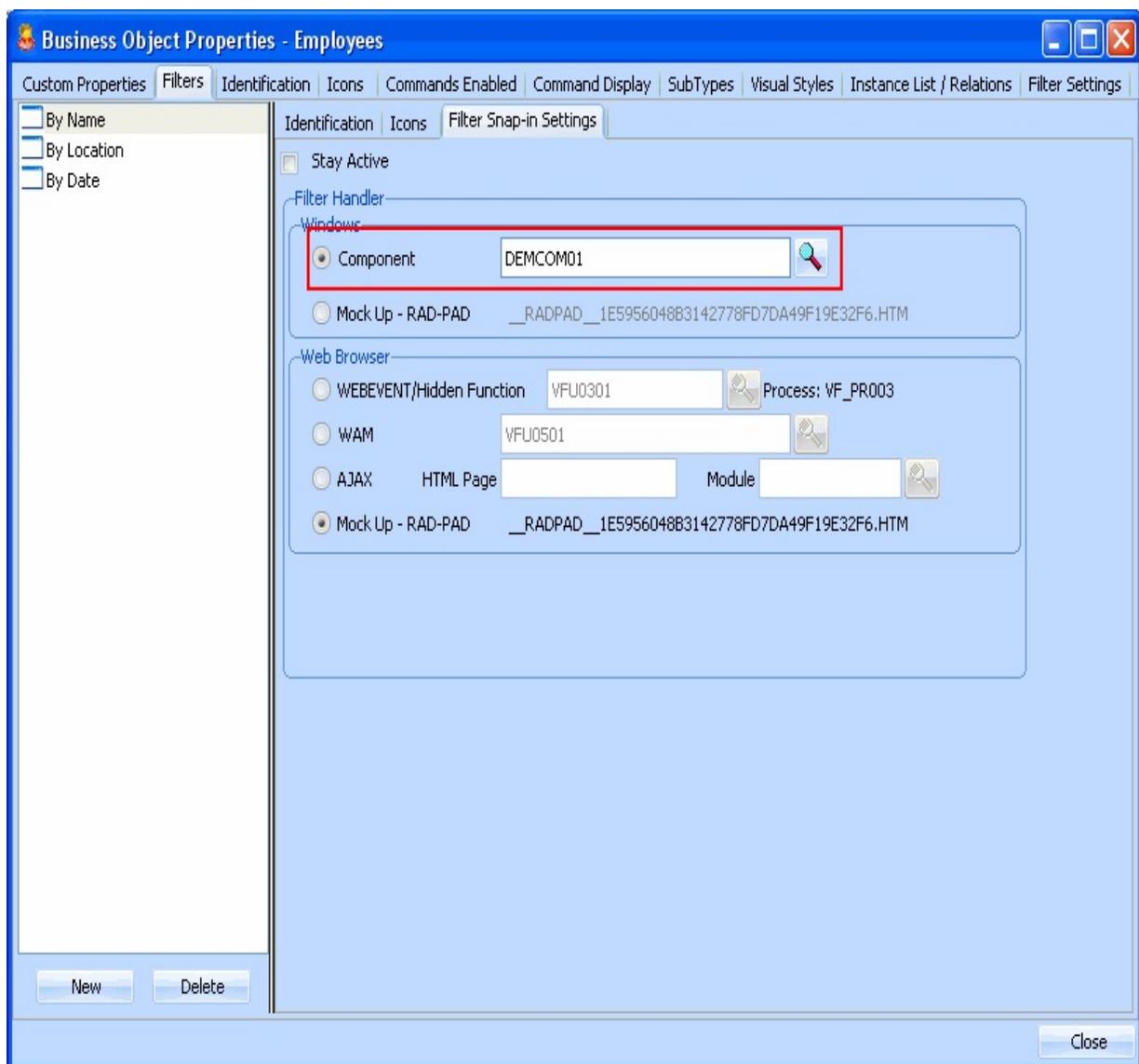
editor.

16. Compile the component.

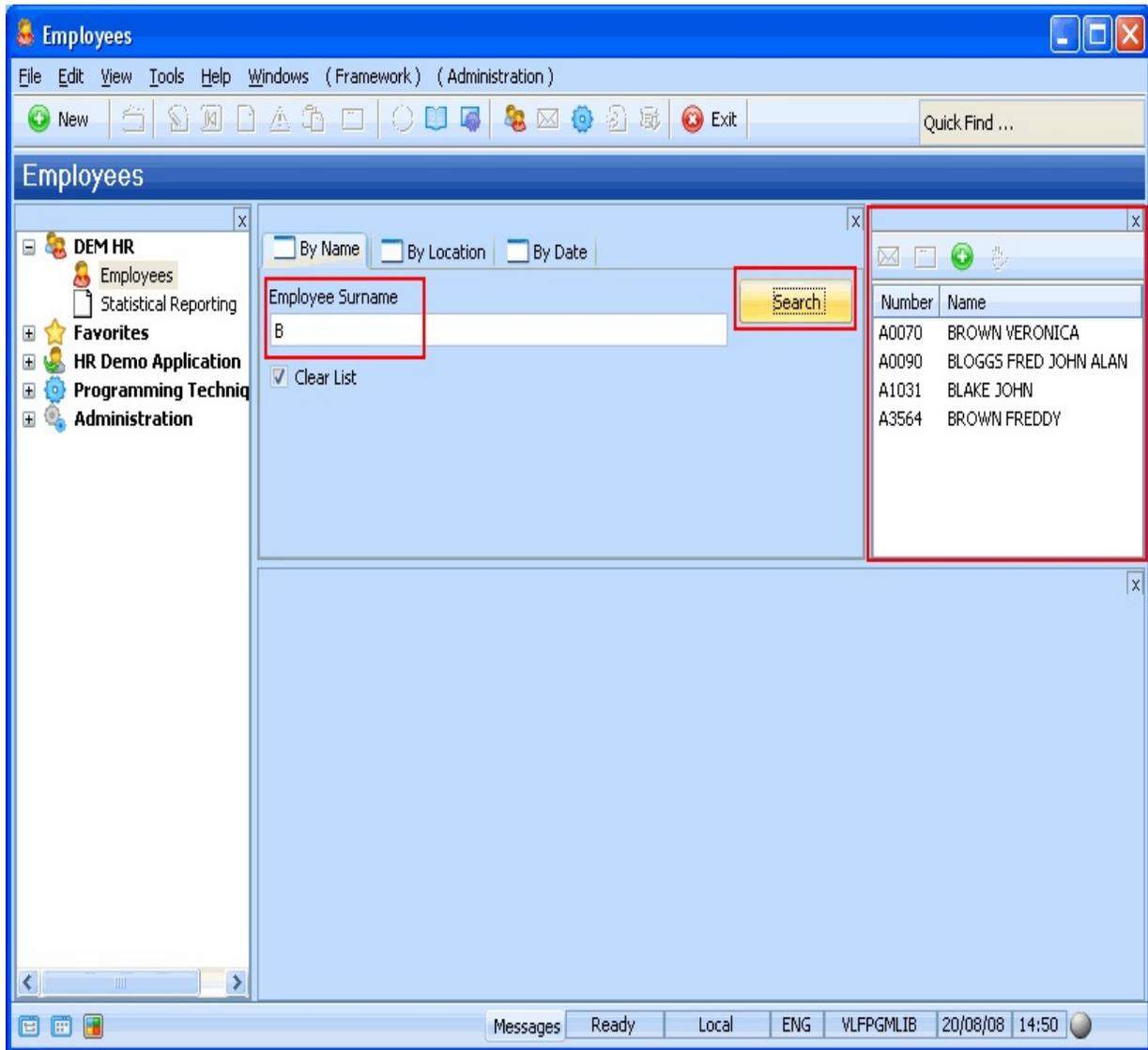
Step 2. Snapping In the By Name Filter

Now that you have compiled your new reusable component (filter) and are ready to test it, you need to snap it into the Framework.

1. In the Framework, close the Program Coding Assistant.
2. Double-click on the Employees business object to display its properties.
3. Display the Filter Snap-in Settings tab.
4. Specify iiiCOM01 as the Windows filter handler component.



5. Close the Employees business object properties and display the By Name filter. You can now see your real filter.
6. Type in a letter in the Surname field and click on the Search button to verify that your real filter has been snapped in the Framework and is usable.



Step 3. Filter Code

Even though you can create most filters simply by using the Program Coding Assistant, you should understand how they are coded.

1. Switch to the Visual LANSAs editor where the iiiCOM01 component is open.
2. Review the generated source code in the Source tab to see how the filter is coded to add data to the instance list:

The Framework is notified that an update is about to occur.

```
Invoke #avListManager.BeginListUpdate
```

Next, the list is cleared of any existing items.

```
Invoke #avListManager.ClearList
```

Next, data is selected. You can use one of the techniques you learnt in the Visual LANSAs Fundamentals tutorials to do this. For example:

```
Select Fields(#XG_Ident) From_File(PSLMST2) With_key(#XG_Keys)  
Generic(*yes) Nbr_Keys(*Compute)
```

Next, the visual identifiers are set up:

```
Change #UF_VisID1 #EMPNO
```

```
Change #UF_VisID2 #SURNAME
```

Then the data is added to the list.

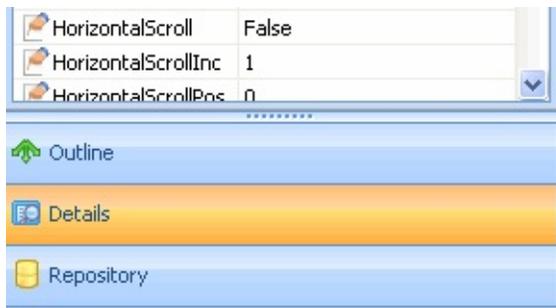
```
Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)  
Visualid2(#UF_VisID2) AKey1(#EMPNO)
```

VisualId1 will be shown in column one of the instance list and VisualId2 will be shown in column two of the instance list. Akey1 is the key that uniquely identifies an employee (in this case the field is alphanumeric, so its Akey1, not Nkey1).

Finally, the Framework is notified that the instance list update is complete.

```
Invoke #avListManager.EndListUpdate)
```

3. Next click on Details tab in the editor to display the properties of your component.

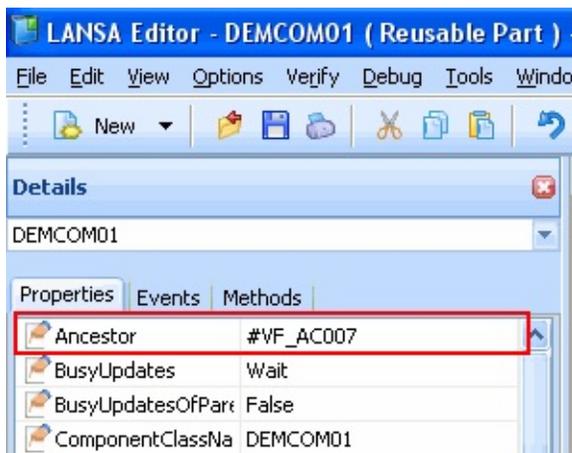


You need to ensure that all properties are displayed:

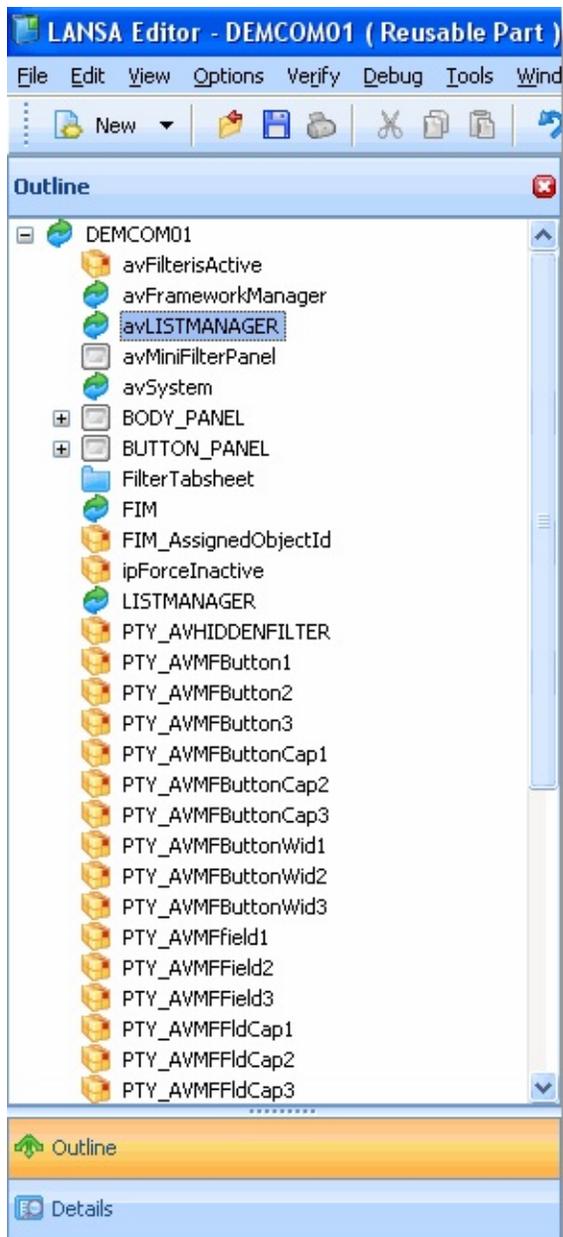
4. Select the Settings option in the Options menu.
5. Click on Details and make sure the Show Advanced Features option is selected.



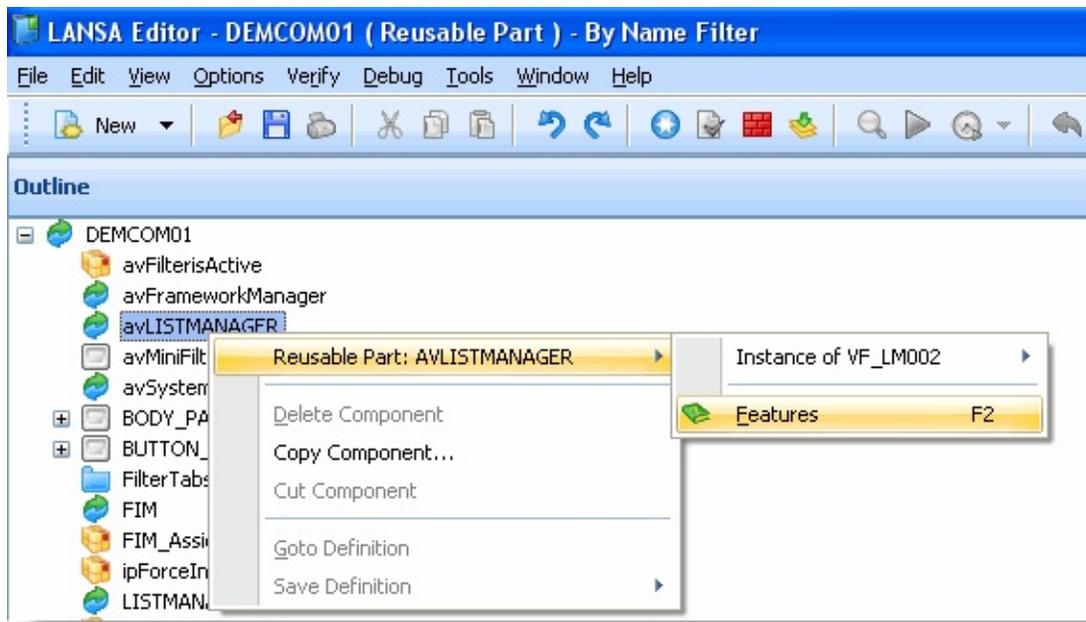
6. Notice that the Ancestor property of the component is #VF_AC007. All filters inherit from this base class which provides a set of predefined behavior.



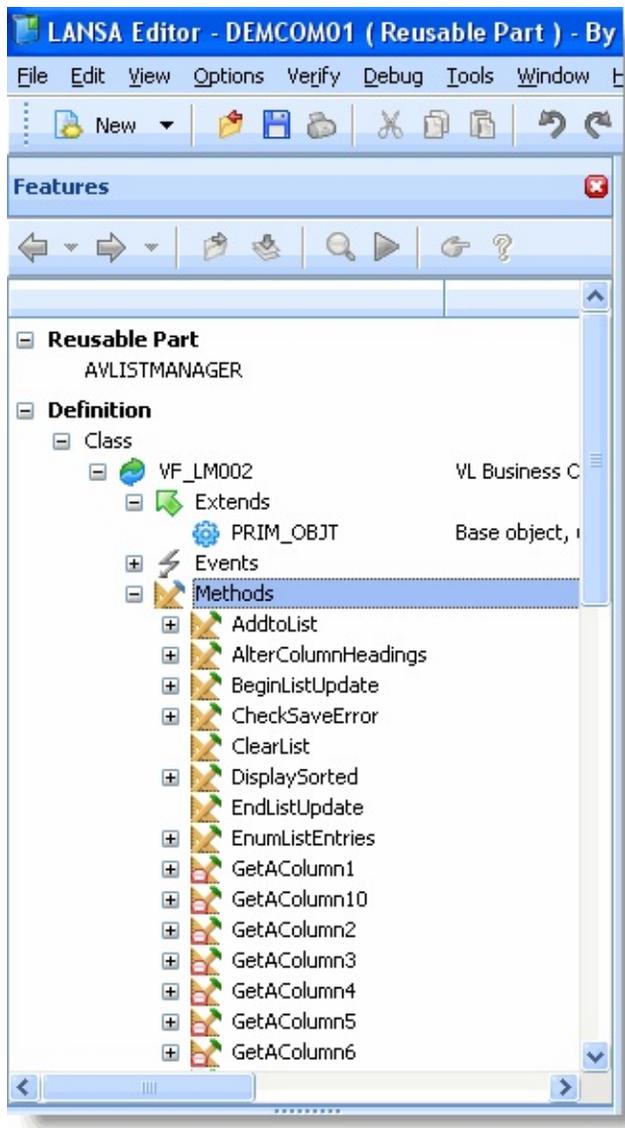
7. Click on the Outline tab in the editor to see what components you inherit from the VF_AC007 ancestor component.



8. Right-click the avLISTMANAGER component and select the Features option.



9. Expand the methods of the component and examine them.



10. Close the iiiCOM01 component.

You may want to read [Windows Filter and Command Handler Anatomy](#) to see how these components are structured.

Step 4. Creating a Real By Location Filter

In this step you create a real By Location filter.

1. In the Framework start the Program Coding Assistant.
2. Drill down through the tree to find your by Location filter and select it.
3. Select Native MS Windows and the type "Filter that searches using a file or view".
4. Specify PSLMST as the physical file.
5. Press the Next button.
 - For your VISUAL IDENTIFIERS specify fields EMPNO, GIVENAME and SURNAME
 - For your PROGRAMATIC IDENTIFIERS specify field EMPNO only.
 - No ADDITIONAL COLUMNS should be specified.
6. Click the Next button to move the Program Coding Assistant forward to the next prompt. This prompt is asking you to select the file or view that the filter should use for searching.
 - Specify PSLMST as the underlying physical file.
 - Select the file view named PSLMST1 (Personnel by Department, Section, Employee Number).
 - Select the search keys DEPARTMENT and SECTION.
 - Uncheck "User must specify all Chosen Keys".
 - Uncheck "Allow Generic Searching".
 - Check "Remember key values between filter executions".
 - Check "Allow user to clear instance list".
7. A screen with additional options is displayed. Do not select any. Click the "Generate Code" button. The right hand side of the Program Coding Assistant now shows the code that it has generated for your filter.
8. In the Generated Code window specify iiiCOM02 as the name of your new filter and give it a description. Then click the Create button to create your filter.

(Alternatively you can copy the generated code to the clipboard by clicking the "Copy Code to Clipboard" button and paste the code into an existing

reusable part).

9. Your filter is displayed in the Visual LANSA editor. Compile it.

Step 5. Snapping in the By Location Filter

In this step, you will snap in your By Location filter.

1. In the Framework close the Program Coding Assistant.
2. Select the iii HR application and double click on the Employee business object.
3. On the resulting Business Object Properties dialog, click on the Filters tab.
4. Select the By Location filter.
5. Click on the Filter Snap-in Settings Tab.
6. Specify iiiCOM02 as the Windows filter handler component.
7. Your filter is now snapped into the Framework and usable.

Summary

Important Observations

- With snap-in real filters you have now created real functionality in your application.

Tips & Techniques

- The source code for the filters used in the demonstration application can be found in the repository in components named DF_*

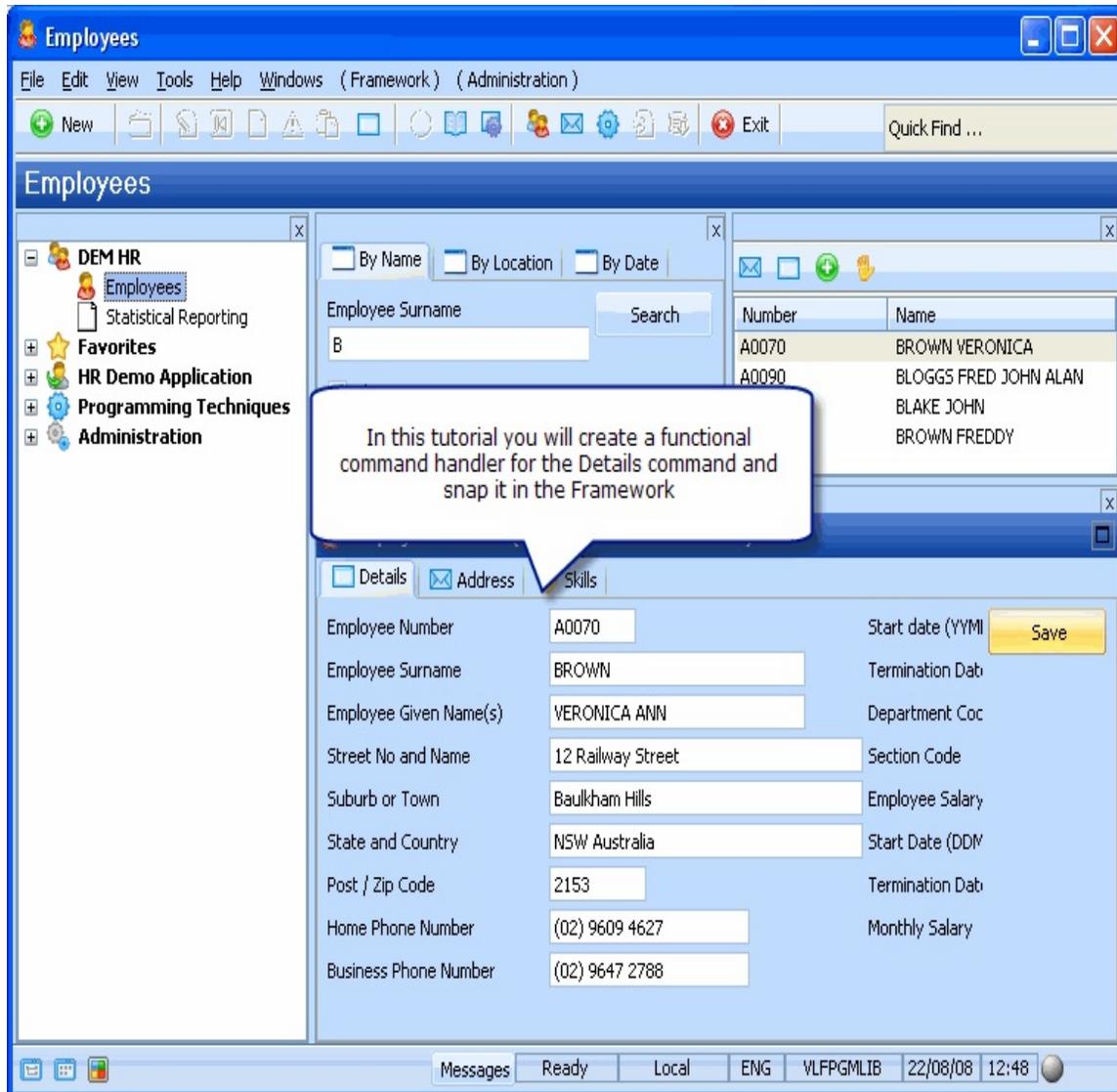
What I Should Know

- What you need to do to create your own filters.
- How you snap them in the Framework.
- How to use the Program Coding Assistant.



VLF007WIN - Snapping in A Real Windows Command Handler Objective

- Learn how to replace prototype command handlers with real handlers which will perform actual processing.
- To replace the Details prototype command handler with a real command handler.



To achieve this objective, you will complete the following steps:

- [Step 1. Creating Your Real Command Handler](#)
- [Step 2. Snapping in Your Command Handler](#)

Before You Begin

You may wish to review:

- [Command](#) in [Key Concepts](#)
- [Command Handler](#)
- [Framework Programming](#).

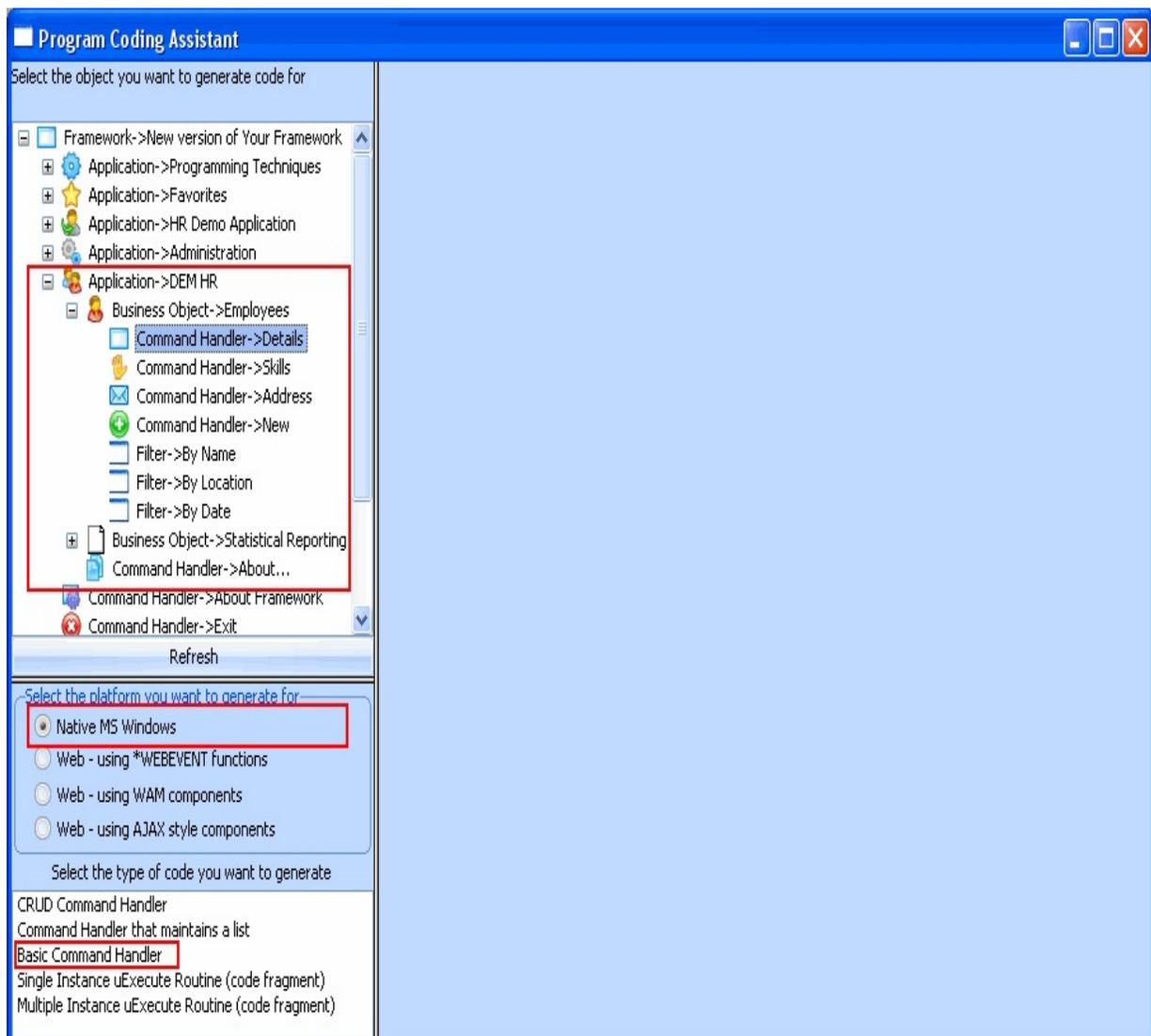
In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)
- [VLF006WIN - Snapping in A Real Windows Filter](#)

Step 1. Creating Your Real Command Handler

In this step, you will create a real command handler for the Details command.

1. Start the Program Coding Assistant.
2. In the list on the top left of the Program Coding Assistant window, select the HR application and then the Details command handler.
3. Select Native MS Windows as the platform and Basic Command Handler as the type of code.



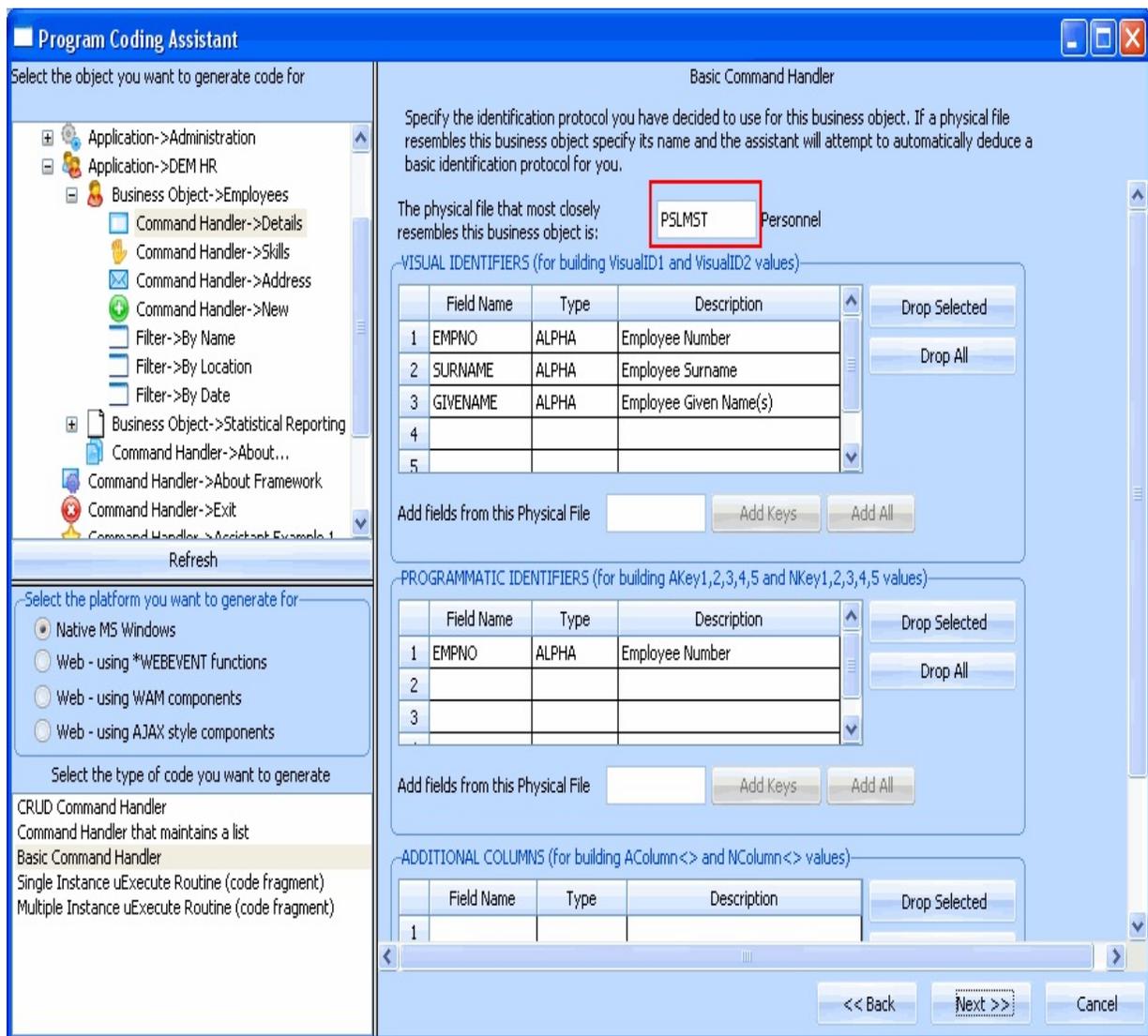
The Basic Command Handler is the most commonly used assistant as you

typically want to create a command handler that displays your data. You then customize the page to meet your specifications.

The CRUD Command Handler is used in conjunction with the CRUD filter and only if the commands defined for the business object are New, Details, Copy, Delete.

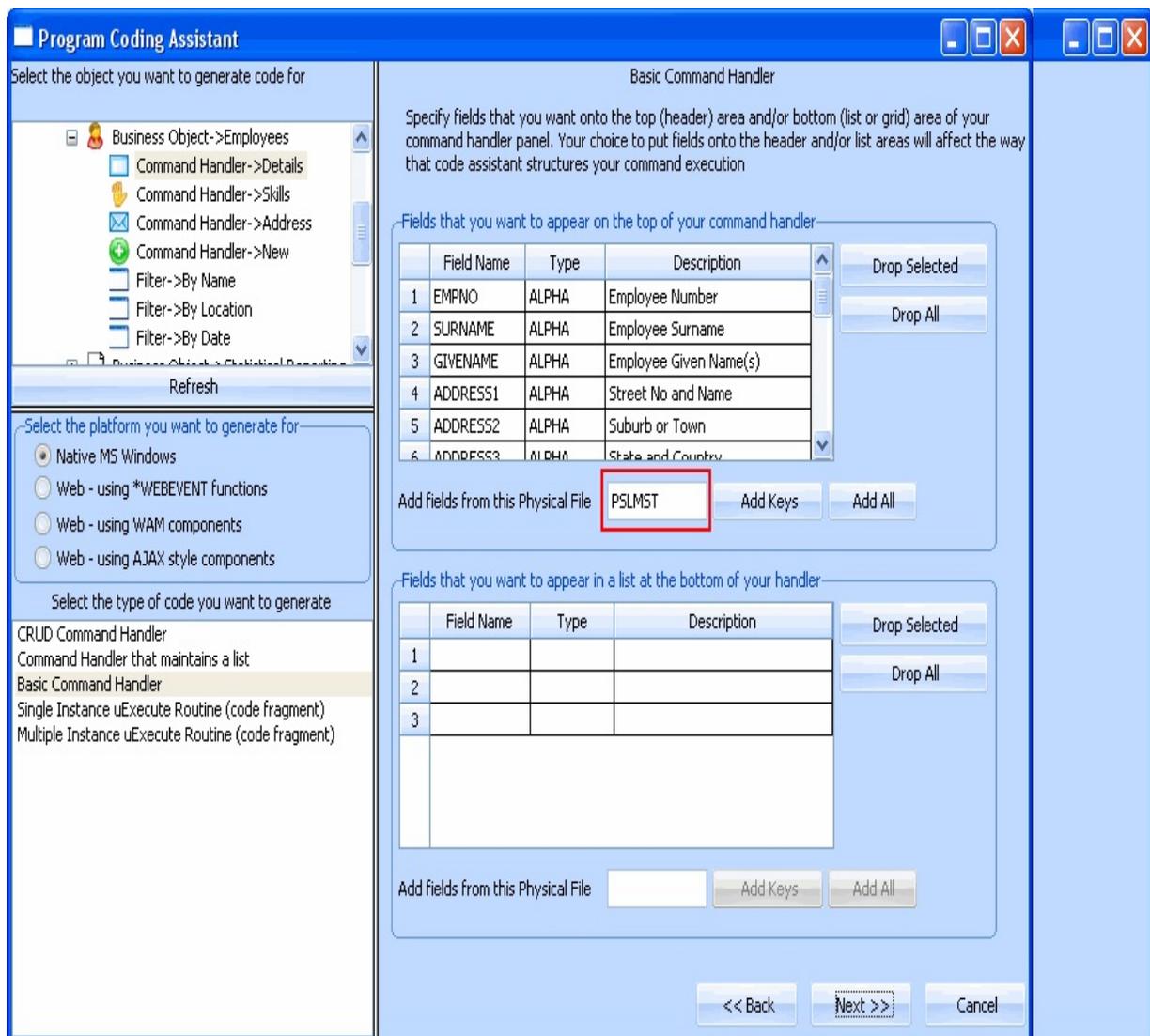
The Command Handler that maintains a list allows you to generate code so that you can use just one command handler to view the details of the instance and a list of information about related data.

4. Click the Next button.
5. On the next page specify PSLMST as The physical file that most closely resembles this business object.



The Program Coding Assistant detects the Visual and Programmatic Identifiers required.

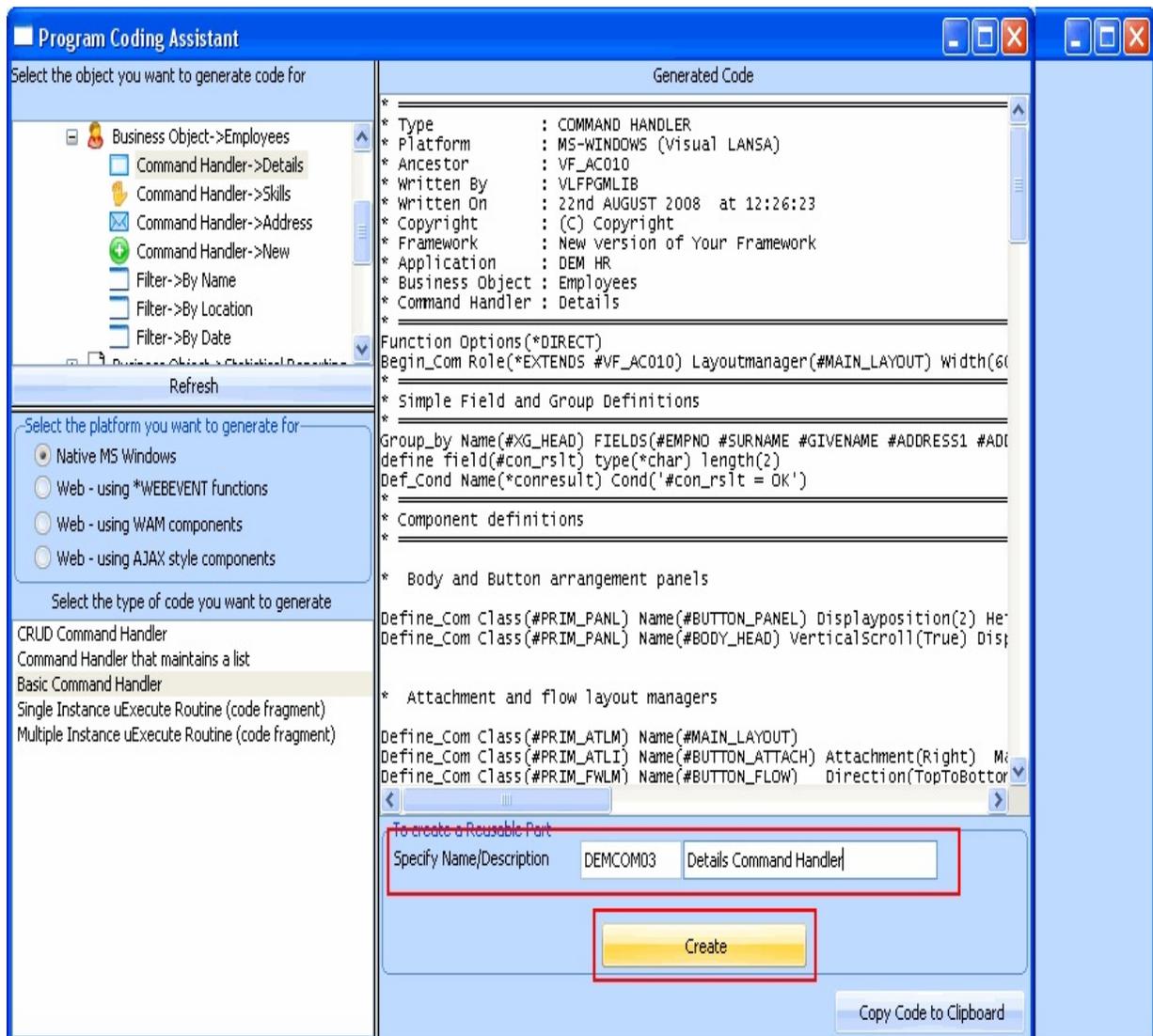
6. Click the Next button.
7. On the next page specify PSLMST in the field Add fields from this physical file in the section Fields that you want to appear at the top of your command handler.
8. Click on the Add All button.



9. Click Next.
10. On the next page click the Generate Code button.

The next page, Generated Code, displays the source code for your command handler. You now need to create the component that will contain the code:

11. Specify iiiCOM03 as the name of your component (where iii are your initials) and Details Command Handler as the description.
12. Click the Create button to create the component.



After a brief delay the command handler component is displayed in the Visual

LANSA editor.

13. Display the Source code of your component.
14. Locate the SAVE_BUTTON.Click event and add a statement to save any changes you make to the fields on the Details command handler.

```
* -----  
EvtRoutine Handling(#SAVE_BUTTON.Click) Options(*NOCLEARMESSAGES *NOCLEARERRORS)  
  * Check that the connection is still live  
  invoke #avFrameworkManager.avCheckConnection ReturnValue(#con_rslt)  
  If Cond('*conresult ')  
    * <your save logic goes here>  
    UPDATE FIELDS(*ALL) IN FILE(PSLMST)  
  endif  
Endroutine
```

15. Locate the uExecute method. Notice that it calls the #avListManager.GetCurrentInstance method to get the key value of the currently selected item in the instance list and then uses this key value to fetch the details.

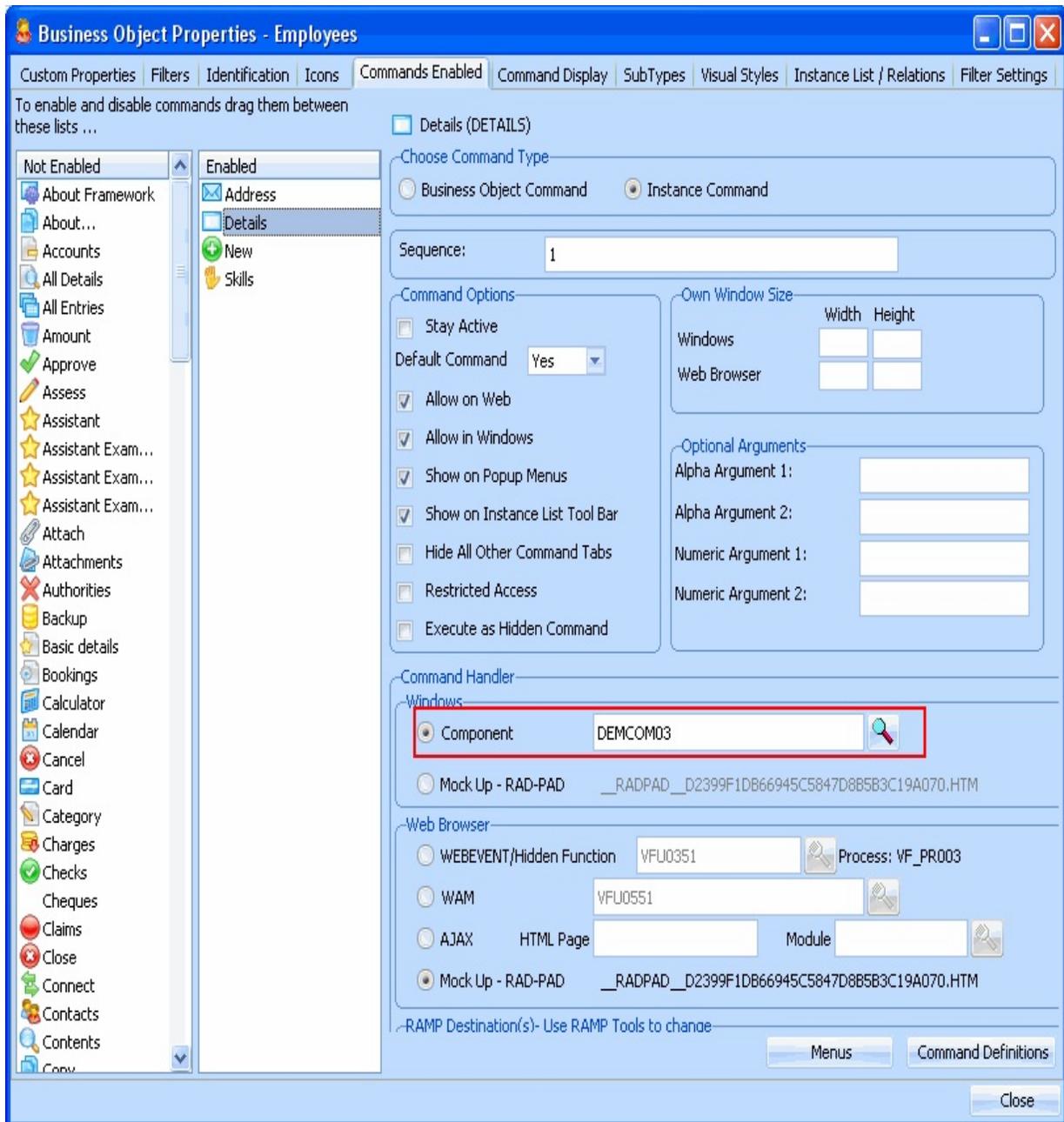
```
Mthroutine Name(uExecute) Options(*REDEFINE)  
  * The return code field and testing condition  
  Define #Ret_Code refld(#IO$STS)  
  Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')  
  * Do any execution logic defined in the ancestor  
  Invoke #Com_Ancestor.uExecute  
  * Check that the connection is still live  
  invoke #avFrameworkManager.avCheckConnection ReturnValue(#con_rslt)  
  If Cond('*conresult ')  
    * Get details of the current instance  
    Invoke #avListManager.GetCurrentInstance Found(#Ret_Code) AKey1(#EMPNO)  
    * Fetch information from the main file to fill in the header fields on the form  
    Fetch Fields(#XG_HEAD) From_File(PSLMST) With_Key(#EMPNO)  
    * ??? Addition logic may be required here ???  
  endif  
Endroutine
```

16. Compile your component.

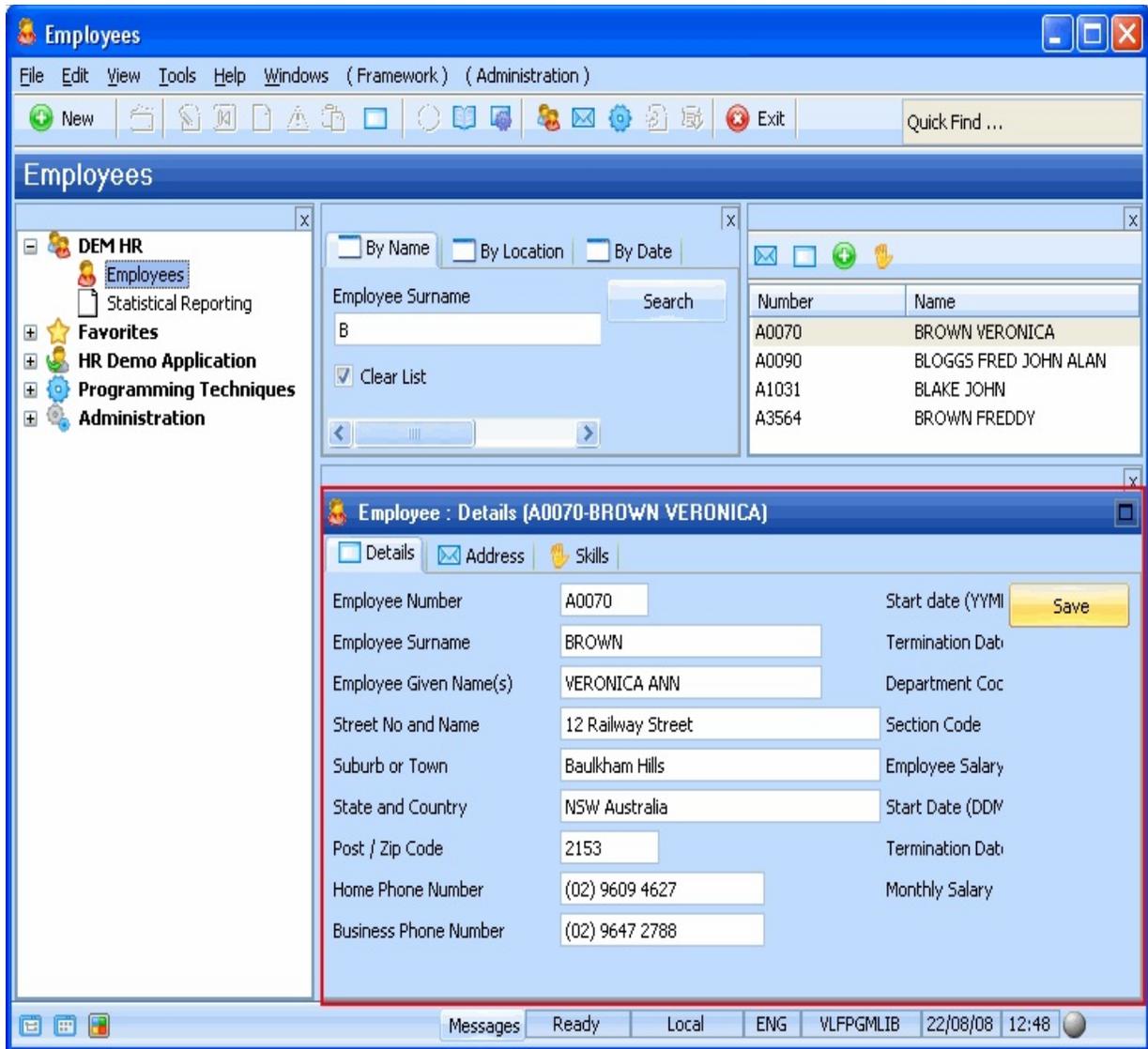
Step 2. Snapping in Your Command Handler

Now that you have compiled your new reusable component (i.e.: your Command Handler) and are ready to test it you need to snap it into the Framework.

1. Display the Framework.
2. Select the iii HR application and display the properties of the Employees object by double-clicking it.
3. On the resulting properties dialog, click on the Commands Enabled tab.
4. Select the Details command.
5. Click on the Component property radio button in the Windows group box.
Type the name of your command handler into the entry field.



6. Close Employee properties. Select the iii HR application and the Employees business object. Click on the Search button to populate the instance list. Then select one item in the instance list to bring up the instance commands.
7. Your command handler for Details is now snapped into the Framework and usable.



8. Try making a change to the details of an employee and saving it.

Summary

Tips & Techniques

- To understand how the command handler interacts with the instance list, read Filter and Command Handler Programming.
- The source code for the command handlers used in the demonstration application can be found in the repository in components named DF_*

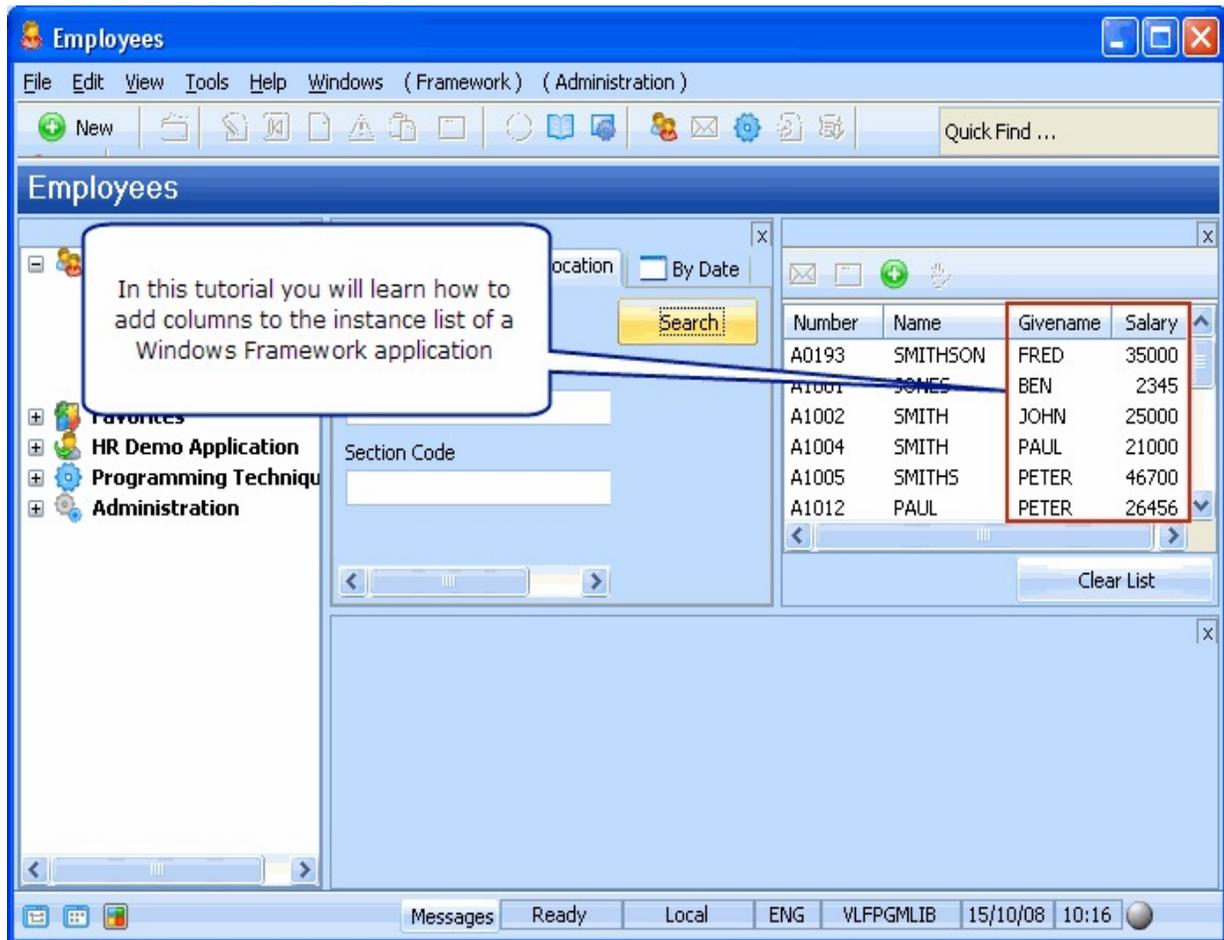
What I Should Know

- What you need to do to create your own command handlers.
- How you snap them in the Framework.
- Filters and Command Handlers are just Reusable Parts which you can customize. However, you can see that up to this point you can get a functional application simply using the Program Coding Assistant without very much coding
- To use the Framework you need to understand VL. However, the level of detail that you must understand is greatly reduced. Creating your own framework to deliver this style of application requires detailed OO knowledge and can take a long time to produce. The VLF allows you to rapidly prototype and deploy an application with no OO knowledge required.

VLF009WIN - Adding Instance List Columns in Windows Applications

Objective

- Learn how to add columns to an [Instance List](#) in a Windows application.



Note: in this tutorial, you will modify the By location filter. Normally, you should do the same modifications to the By name filter.

To achieve this objective, you will complete the following steps:

- [Step 1: Add Columns to the Instance List](#)
- [Step 2: Change your filter](#)
- [Step 3: Remove the Additional Columns.](#)

Before You Begin

You may wish to review:

- [List Manager](#)

- [Adding Additional Columns to Instance Lists](#) .

In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)
- [VLF006WIN - Snapping in A Real Windows Filter](#)
- [VLF007WIN - Snapping in A Real Windows Command Handler.](#)

Step 1: Add Columns to the Instance List

In this step, you will configure your Employee business object to make the extra columns visible in the instance list.

1. Start the Framework as a designer.
2. Open the properties of the Employees business object.
3. Display the [Instance List/Relations](#) tab sheet.
4. Two visual identifiers are already defined. Add two additional columns:

Column Sequence Column Type Column Caption

30	ACOLUMN1	Givename
40	NCOLUMN1	Salary

The column definitions now look like this:

Sequence	Type	Caption	Width % (Total 25%)	Decimals	Edit Code	Date/Time Output Format
10	VISUALID1	Number	25		Default	SYSFMT8
20	VISUALID2	Name			Default	SYSFMT8
30	ACOLUMN1	Givename			Default	SYSFMT8
40	NCOLUMN1	Salary			Default	SYSFMT8
	ACOLUMN2				Default	SYSFMT8
	ACOLUMN3				Default	SYSFMT8
	ACOLUMN4				Default	SYSFMT8
	ACOLUMN5				Default	SYSFMT8

Step 2: Change your filter

Finally, you need to make some changes to your filter to fill the new instance list columns with data.

1. Close the Framework
2. Open the source of the By Location filter (reusable part iiiCOM02) which you created in [VLF006WIN - Snapping in A Real Windows Filter](#).
3. Make these changes to the code:

Change the GROUP_BY command to include the #SALARY field:

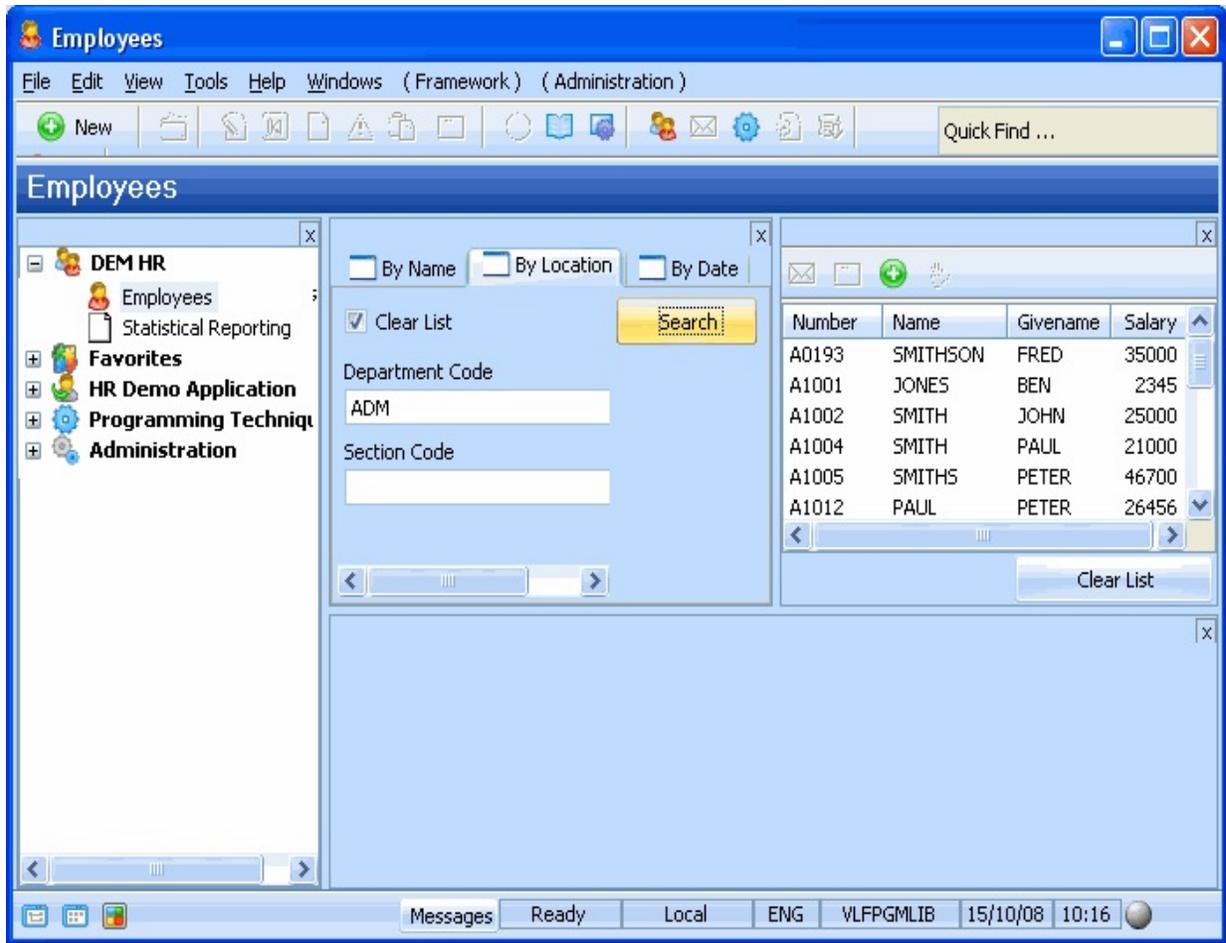
```
Group_By Name(#XG_Ident) Fields( #EMPNO #GIVENAME #SURNAME
```

Locate the following statement Select Fields(#XG_Ident) command and change the AddtoList statement to:

```
* Add instance details to the instance list
```

```
Invoke #avListManager.AddtoList Visualid1(#EMPNO) AKey1(#EMPNO)  
Visualid2(#Surname) AColumn1(#Givename) NColumn1(#Salary)
```

4. Compile the reusable part.
5. Start the Framework and test the result.



6. Close the Framework.

Step 3: Remove the Additional Columns

In this step you will remove the additional columns.

1. Display the properties of the Employees business object.
2. Display the Instance List / Relations tab.
3. Remove the column sequence numbers for Givenname and Salary fields.
4. Close the properties and save the Framework.

You do not need to change the filter because you will replace it with a mini filter in the following tutorial.

Summary

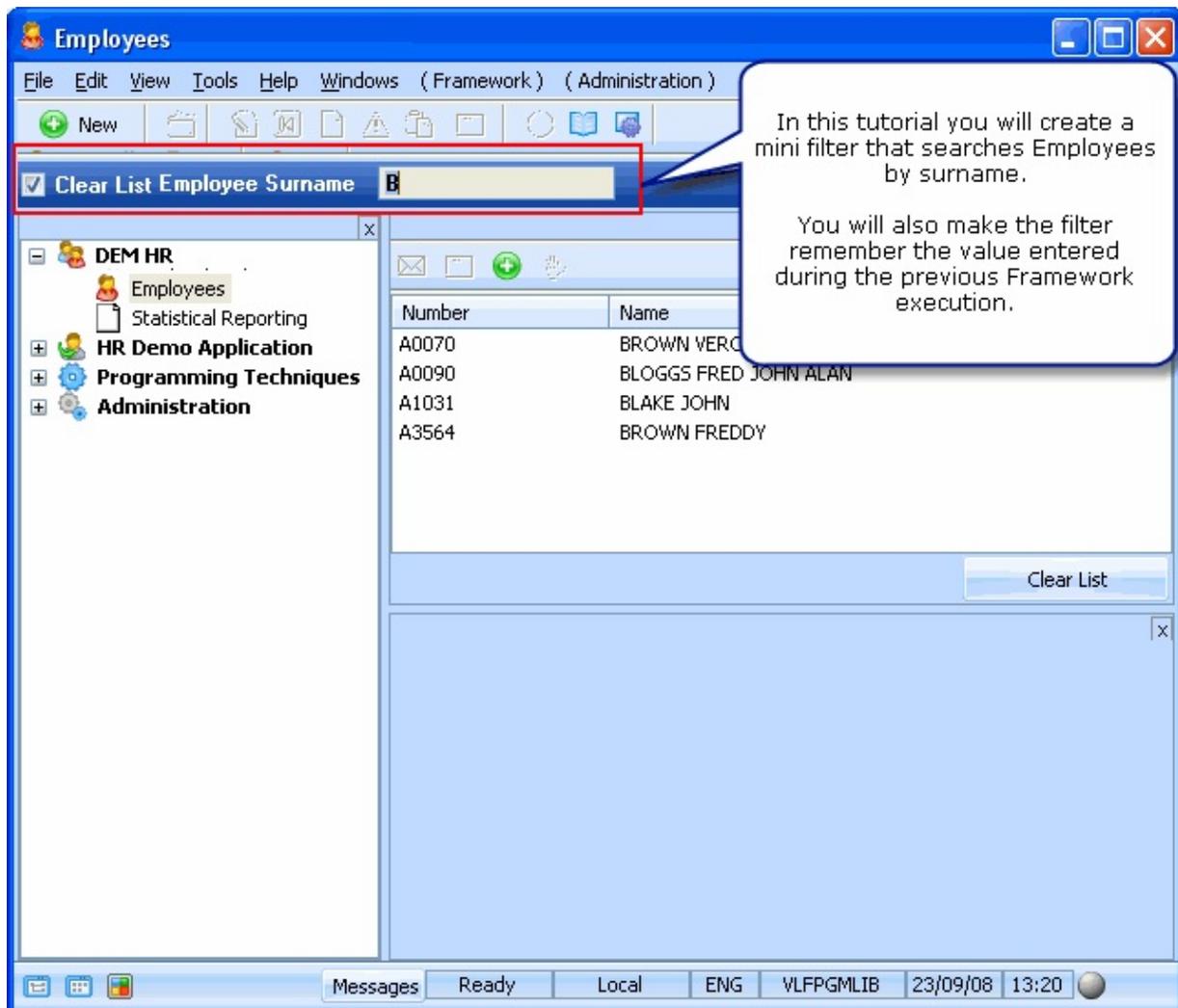
What You Should Know

How to add columns to a instance list.

VLF010WIN - Creating a Mini Filter

Objectives

- To learn how to create [Mini Filters](#).
- To understand how you can use [The Virtual Clipboard](#) to remember values between Framework executions.



To achieve this objective, you will complete the following steps:

Step 1. [Create the Mini Filter Interface](#)

Step 2. [Write the Mini Filter Code](#)

Step 3. [Snap in the Mini filter and Test It](#)

Step 4. [Use the Virtual Clipboard to Save and Restore the Search Value](#)

[Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

- Tutorials VLF000 – VLF007WIN and VLF009WIN

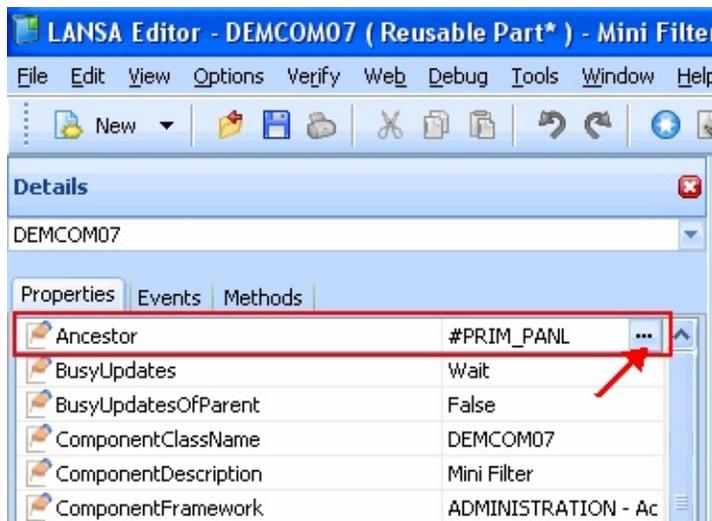
Step 1. Create the Mini Filter Interface

In this step you will create a reusable part that will become a mini filter for the Employees business object.

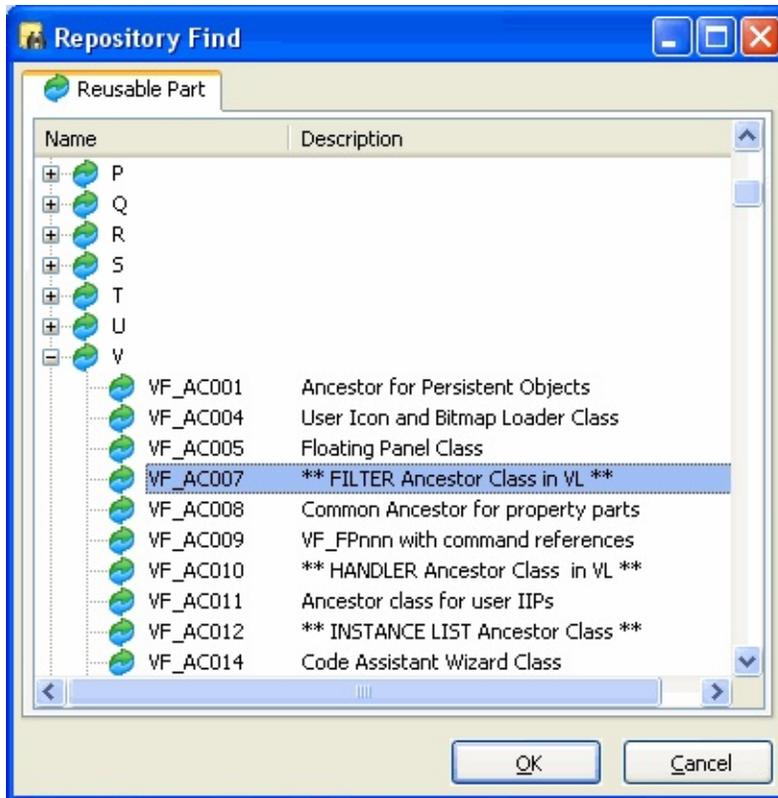
1. Start the Visual LANSA editor.
2. Create a reusable part. Specify iiiCOM07 as the name of your filter and Employee Mini Filter as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).

You need to ensure that all properties are displayed:

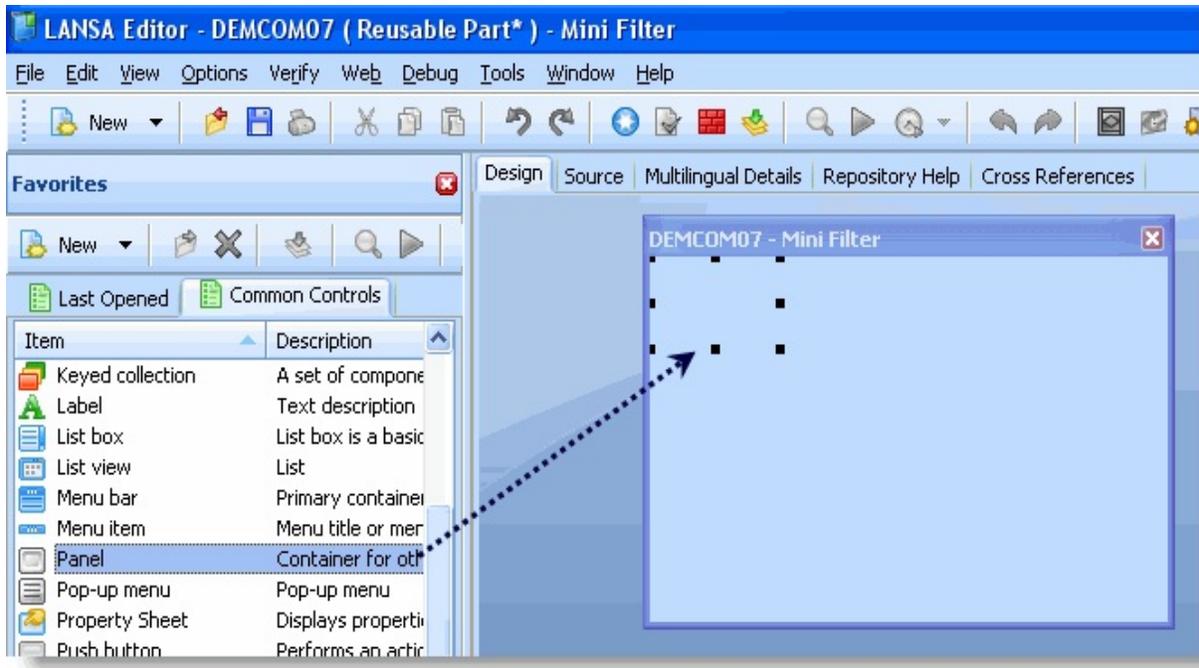
3. Select the Settings option in the Options menu.
4. Click on Details and make sure the Show Advanced Features option is selected.
5. Click on the Details tab of the component.
6. Click on the Ancestor property and click the prompt button:



7. Select the component #VF_AC007. It is the ancestor component for filters from which your mini filter will inherit properties, events and methods.



8. Click on the OK button.
9. Click on the Favorites tab and select the Common Controls tab. (If you are using an earlier version of Visual LANSA than 11 SP 5, locate the controls in the Primitives group under System Information in the Repository tab.)
10. Drag the Panel control to the Design view and drop it into the upper left part of the reusable part.



The panel will contain the contents of your mini filter. At run time you will tell the filter which panel contains the contents to be displayed in the mini filter area. Therefore, you could have multiple mini filter panels and display the appropriate one at run time.

11. Click on the Details tab with the panel selected.

12. Set the panel properties:

Left	2
Top	2
Width	400
Height	25

13. Display the Favorites tab and the Common Controls tab on it.

14. Drag a check box control to the upper left corner of the panel.



You must place all your objects in the upper left portion of the panel. The maximum height of your objects on the panel is restricted to the height of the tool bar in the Visual LANSAs Framework Application which is 25 pixels. Therefore, if you add the value in the Top property of your component to the value in the Height property of your component, the sum should not exceed 25. The check box will be used by the end-user to optionally clear the instance list when using the filter.

15. In the Details tab, set the default properties for the check box:

ButtonState	Checked
Caption	Clear List
Height	21
Left	0
Name	CLEAR_LIST
Top	0
Width	75

16. Click on the Repository tab.

17. Locate the PSLMST file and expand it.
18. Drag the SURNAME field to the right of the check box on the panel.



19. Click on the Details tab and set the properties of the SURNAME field:

Left	75
MarginLeft	120
Top	1
Width	250

20. Save your mini filter.

Step 2. Write the Mini Filter Code

In this step you will write the code for your mini filter. You will need to manually code the filter because there is no Program Coding Assistant for mini filters.

1. Display the Source tab of your filter.
2. Add this statement after the component definitions:

```
Def_List Name(#Save_Keys) Fields(#SURNAME) Type(*Working)
Entrys(1)
```

The Save_Keys working list will be used to save the key values from overwrites done by the select loop.

3. Next add a uInitialize method routine.

```
Mthroutine Name(uInitialize) Options(*Redefine)

#COM_OWNER.avMiniFilter := true
#COM_OWNER.avMiniFilterpanel <= #PANL_1

Endroutine
```

- The uInitialize method routine is executed just once when the filter or command handler is being created.
- The avMiniFilter property indicates that the filter should be a mini filter.
- The avMinifilterpanel sets a reference to the panel within the filter. The characters “<=” between two components assign a reference. You can have multiple panels in your filter and use one of many mini filter panels based on a condition rather than always using the same one.

Your code will now look like this:

```

Function Options(*DIRECT)
  BEGIN_COM ROLE(*EXTENDS #VF_AC007) HEIGHT(192) WIDTH(388)
  DEFINE_COM CLASS(#PRIM_PANL) NAME(#PANL_1) DISPLAYPOSITION(1) HEIGHT(49)
    HORIZONTALSCROLL(True) LEFT(2) PARENT(#COM_OWNER) TABPOSITION(1) TABSTOP(False) TOP(2)
    VERTICALSCROLL(True) WIDTH(400)

    Def_List Name(#Save_Keys) Fields(#SURNAME) Type(*Working) Entrys(1)

  Mthroutine Name(uInitialize) Options(*Redefine)
    #COM_OWNER.avMiniFilter := true
    #COM_OWNER.avMiniFilterpanel <= #PANL_1
  Endroutine

```

- Next add an event routine to handle the KeyPress event of the SURNAME field. This event will execute when the Enter key is pressed.

```

EVTROUTINE HANDLING(#SURNAME.KeyPress)
OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES)
KeyCode(#keycode)
if ('#keycode.value = Enter')

endif
ENDROUTINE

```

- If the Enter key is pressed, save the current key values from overwrites done by the select loop:

```
Inz_List #Save_Keys 1
```

- Indicate that the instance list updating is about to start and then clear the instance list if the checkbox has been selected:

```

Invoke #avListManager.BeginListUpdate

If '#Clear_List.ButtonState = Checked'
Invoke #avListManager.ClearList
Endif

```

7. Then select employees that match the search criteria and set up the visual identifiers, then add the entries to the instance list:

```
Select Fields(#EMPNO #SURNAME #GIVENAME #SALARY)
From_File(PSLMST2) With_key(#SURNAME) Generic(*yes)
Nbr_Keys(*Compute)

Use Builtin(BCONCAT) With_Args(#GIVENAME #SURNAME)
To_Get(#FULLNAME)
Invoke #avListManager.AddtoList Visualid1(#EMPNO)
Visualid2(#FULLNAME) NColumn1(#SALARY) AKey1(#EMPNO)

EndSelect
```

8. Lastly indicate that the instance list update is complete and restore the saved key values:

```
Invoke #avListManager.EndListUpdate
Get_Entry 1 #Save_Keys
```

Your finished event routine will look like this:

```
EVTROUTINE HANDLING(#SURNAME.KeyPress) OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES) KeyCode(#keycode)
  if ('#keycode.value = Enter')
    Inz_List #Save_Keys 1
    Invoke #avListManager.BeginListUpdate
    If '#Clear_List.ButtonState = Checked'
      Invoke #avListManager.ClearList
    Endif

    Select Fields(#EMPNO #SURNAME #GIVENAME) From_File(PSLMST2) With_key(#SURNAME) Generic(*yes)
      Nbr_Keys(*Compute)
      DCM0123/Warning : no key specified to match file PSLMST2 from DC@DEMOLIB key field GIVENAME

      Use Builtin(BCONCAT) With_Args(#GIVENAME #SURNAME) To_Get(#FULLNAME)
      Invoke #avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#FULLNAME) NColumn1(#SALARY)
      AKey1(#EMPNO)
    EndSelect

    Invoke #avListManager.EndListUpdate

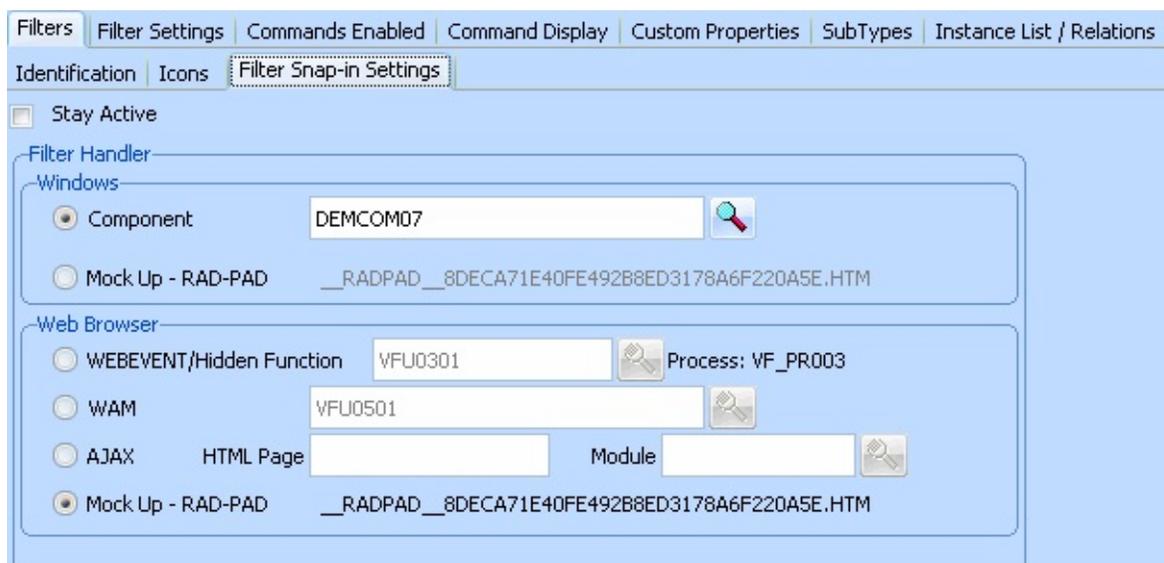
    Get_Entry 1 #Save_Keys
  endif
endroutine
```

5. Compile the filter.

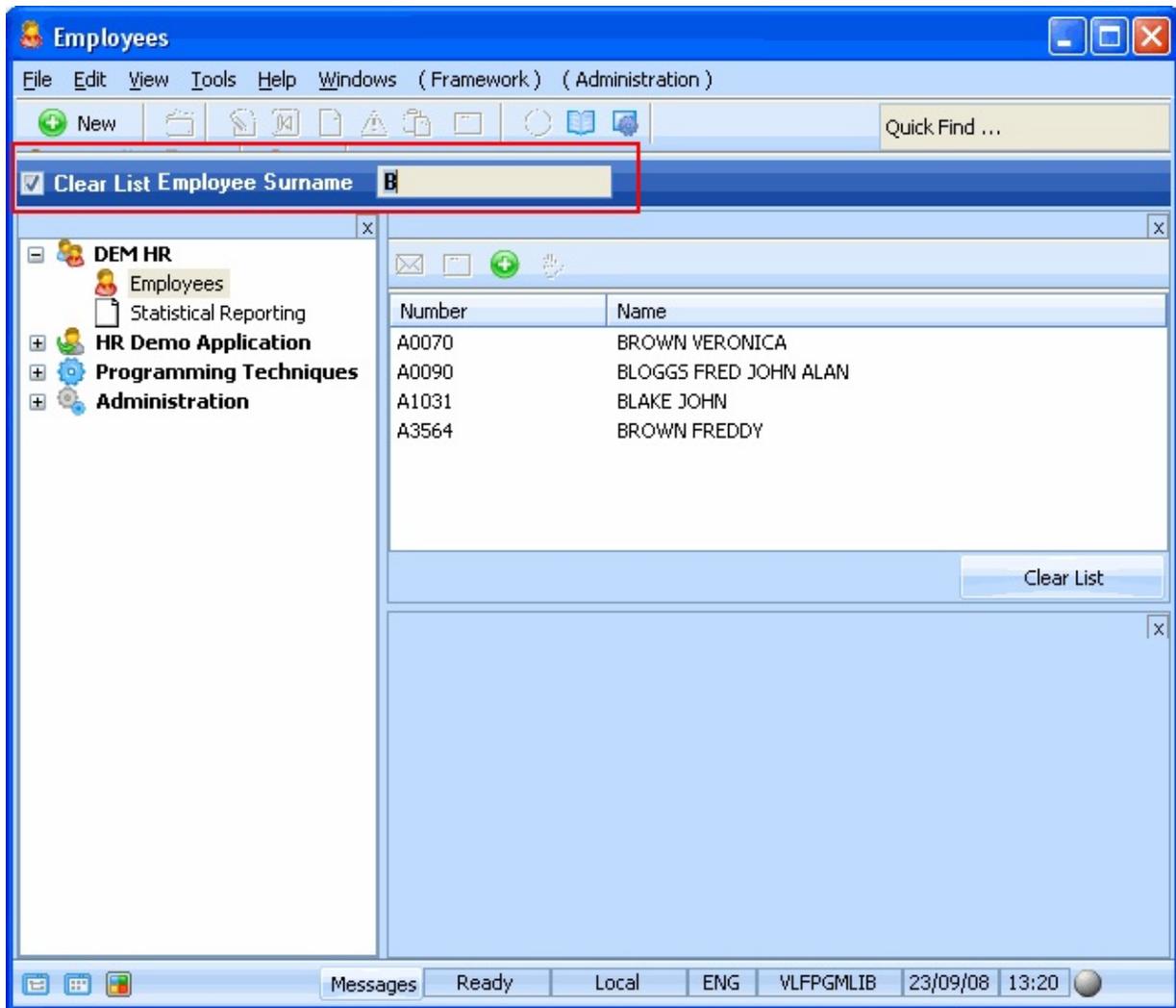
Step 3. Snap in the Mini filter and Test It

In this step you will snap the mini filter in the Employees business object and test it.

1. Start the Visual LANSA Framework.
2. Select the Employees business object and display its properties.
3. Delete the three existing filters. When a mini filter is being used, there can only be one filter for the business object because the reason for having a mini filter is to save space to allow the instance list to be wider.
4. Save and restart the Framework so that the filter deletion is complete.
5. Create a new filter.
6. Specify iiiCOM07 as the snap-in filter.



7. Close the Employee properties.
8. Click on the Statistical Reporting business object.
9. Then click on the Employee business object. You can now see the mini filter.
10. Test the filter by typing in a letter in the SURNAME field and pressing Enter.



Notice how much more screen real estate the instance list now has because the mini filter takes up very little space.

Also note that you can show any control that fits within the mini filter. For example you can add combo boxes, drop downs, check boxes, and do instant editing on the mini filter panel.

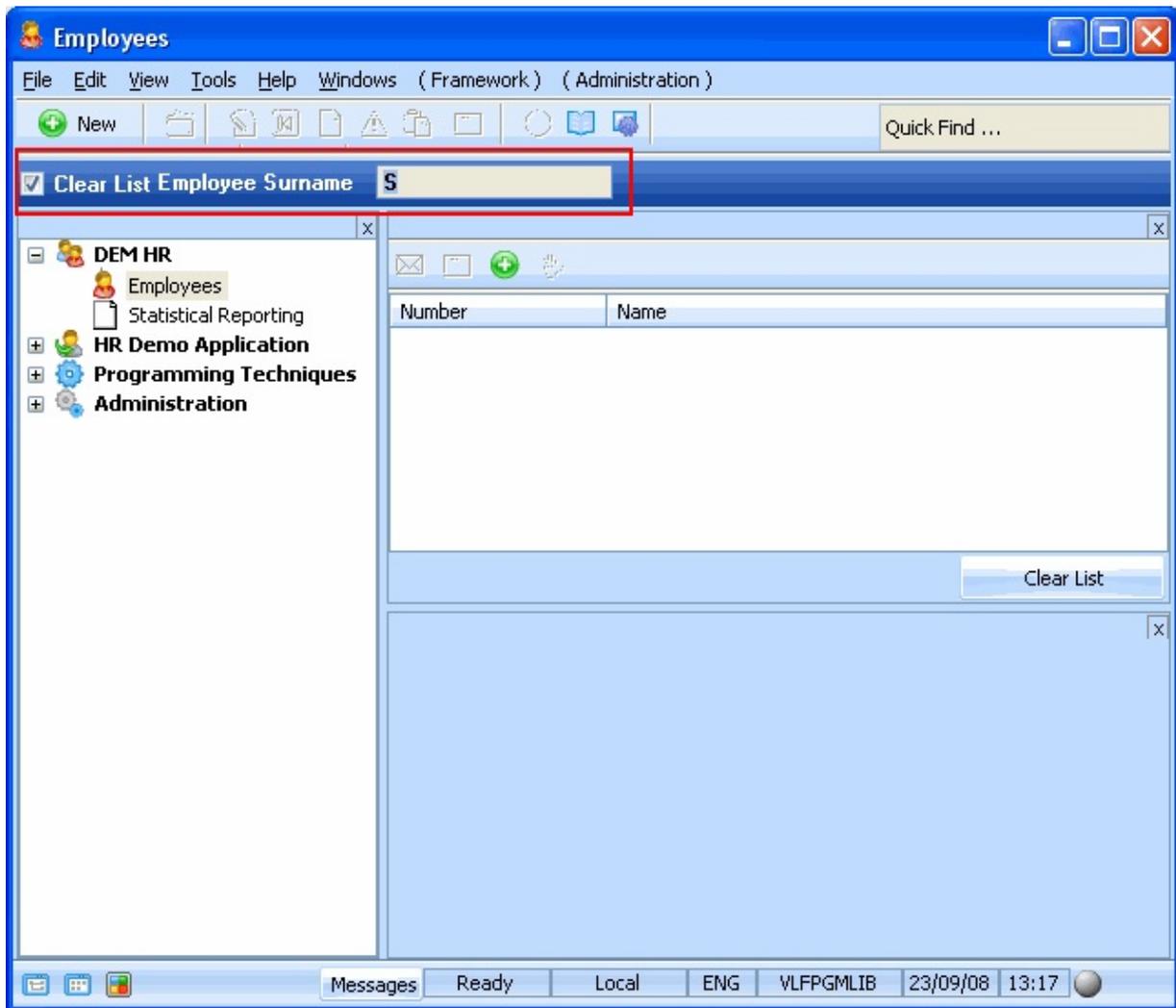
Remember that the maximum height of the tool bar is 25 pixels.


```

Mthroutine Name(uInitialize) Options(*Redefine)
#COM_OWNER.avMiniFilter := true
#COM_OWNER.avMiniFilterpanel <= #BODY PANEL
Invoke #AvFrameworkManager.avRestoreValue WithID1(SURNAME) ToAValue(#SurName)
Endroutine

```

7. Now compile the filter.
8. Start the Framework and locate the Employees business object.
9. Type S in the mini filter and press Enter to fill the instance list.
10. Close the Framework and start it again and select the Employees business object. Notice that the mini filter now contains the letter S which was entered in the previous Framework execution.



Summary

Important Observations

- You can have multiple panels in your filter and use one of many mini filter panels based on a condition rather than always using the same one.
- When a mini filter is being used, there can only be one filter for the business object because the reason for having a mini filter is to save space to allow the instance list to be wider.

Tips & Techniques

- The Advanced section of the Programming Techniques sample application has examples of simple and complex minifilters.

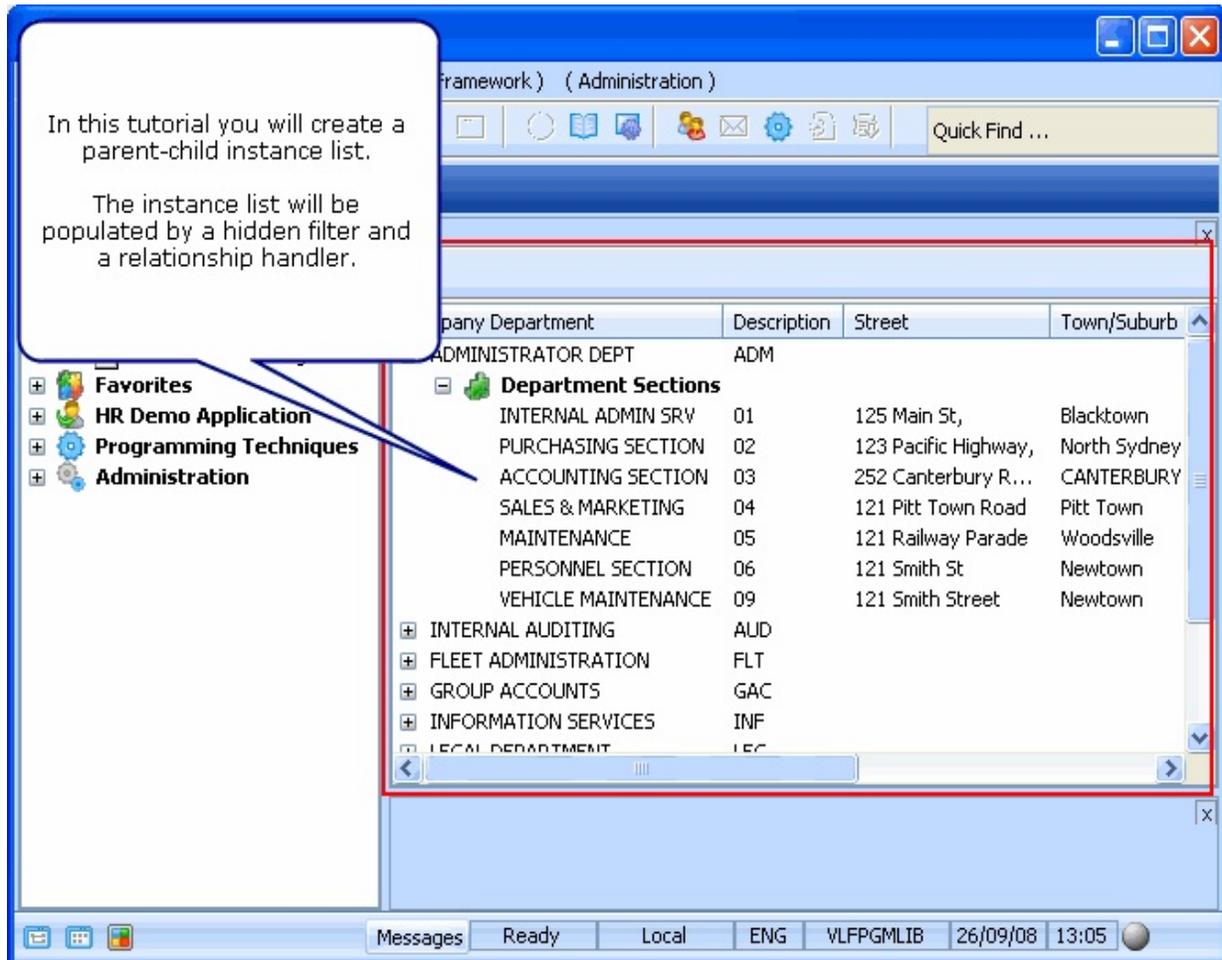
What I Should Know

- How to create a mini filter.
- How to use the virtual clipboard.

VLF011WIN - Creating a Parent Child Instance List

Objectives

- To learn how to create a parent-child instance list (see [Instance Lists with Different Types of Objects](#)) with a [Hidden Filter](#) and a relationship handler.



To achieve this objective, you will complete the following steps:

Step 1. [Create Two New Business Objects](#)

Step 2. [Establish the Parent-Child Relationship](#)

Step 3. [Create a Hidden Filter for Company Departments](#)

Step 4. [Create a Relationship Handler to Load Sections](#)

Step 5. [Display Additional Columns in the Instance List](#)

Step 6. [Access the Properties of Hidden Child Objects](#)

[Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

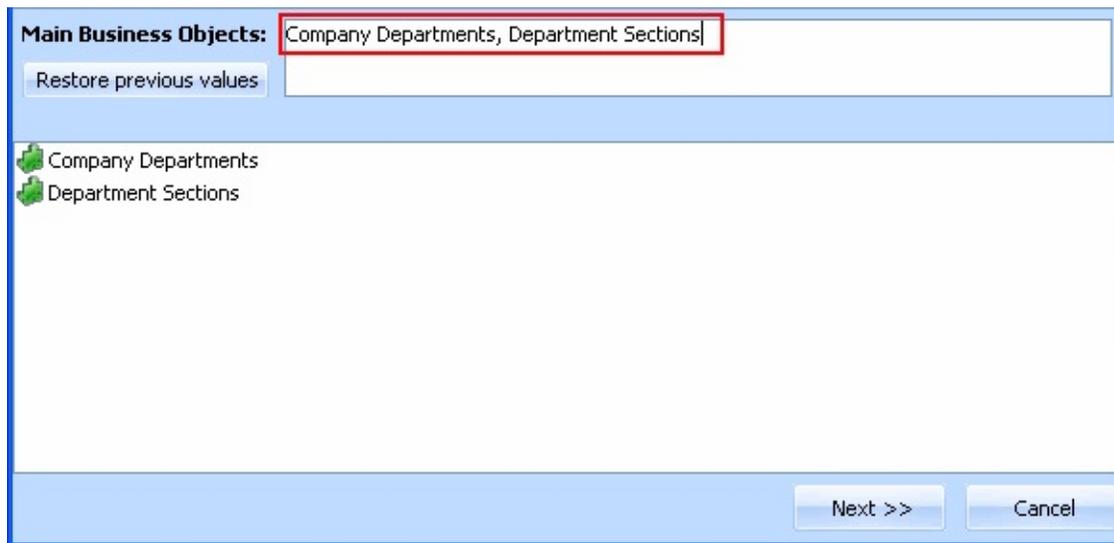
- Tutorials VLF000 – VLF007WIN.

Step 1. Create Two New Business Objects

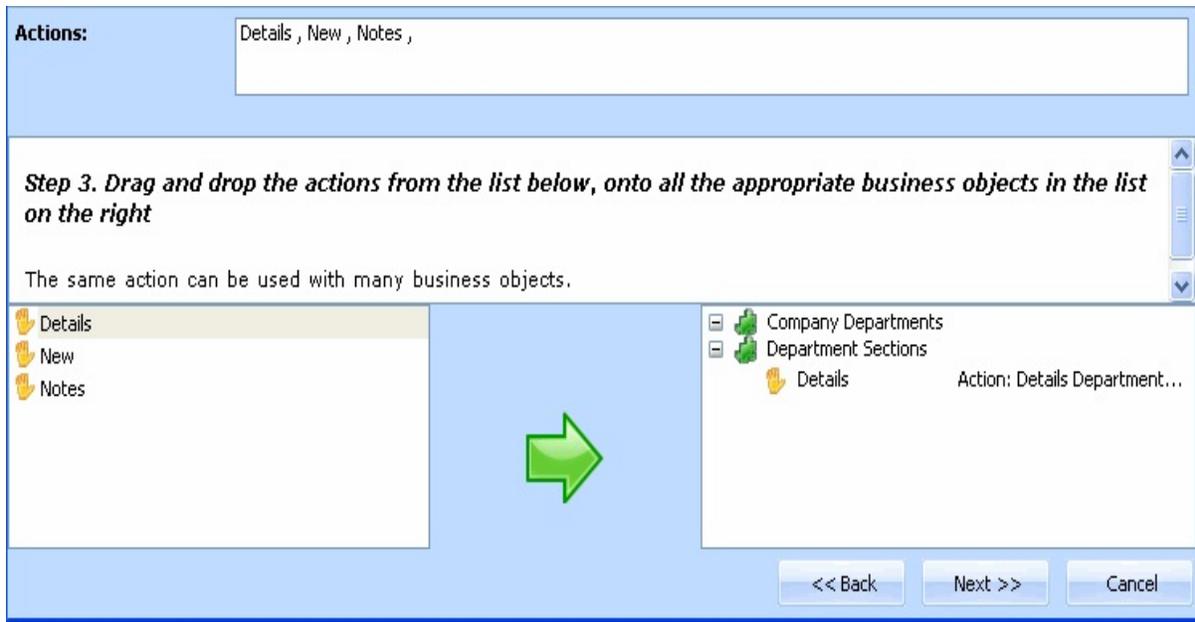
In this step you will use the Instant Prototyping Assistant to create two business objects: Company Departments and Department Sections.

The Sections business object will become the child of the Company Departments object.

1. Start the Instant Prototyping Assistant in the Framework.
2. Type in the names of two new business objects Company Departments and Department Sections. Remember to separate the names with a comma.

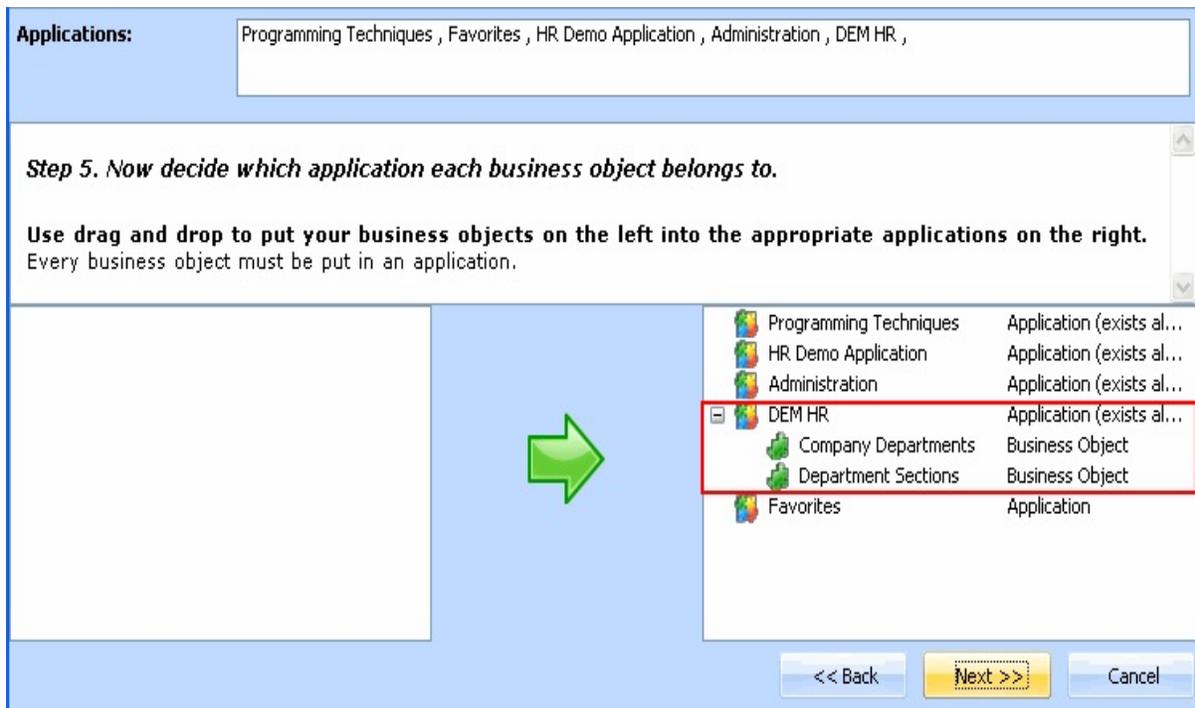


3. Click Next.
4. Drag the Details command to the Department Sections business object.



5. Click Next.

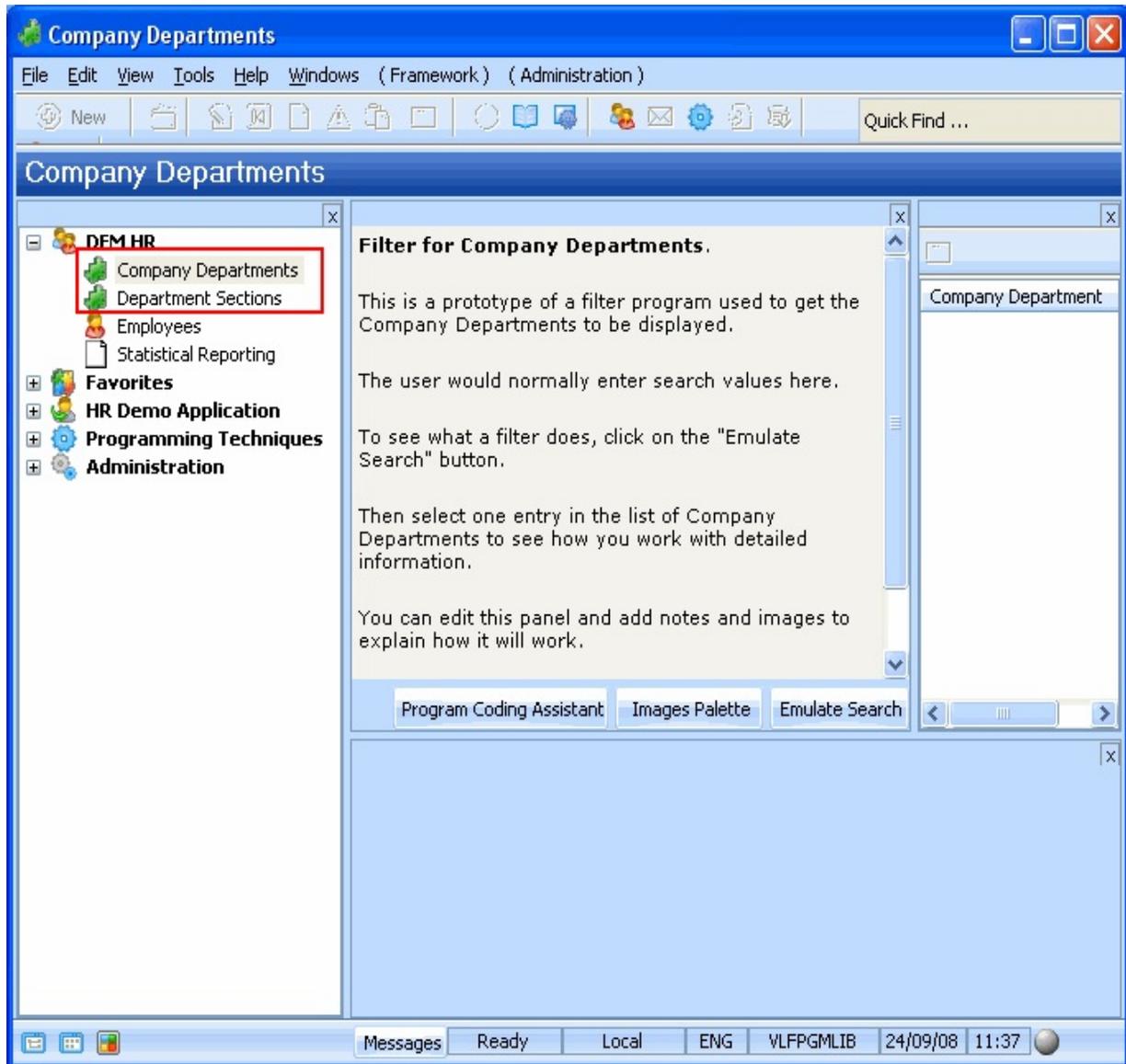
6. Drag both business objects to the iii HR application.



7. Click Next.

8. Click Finish.

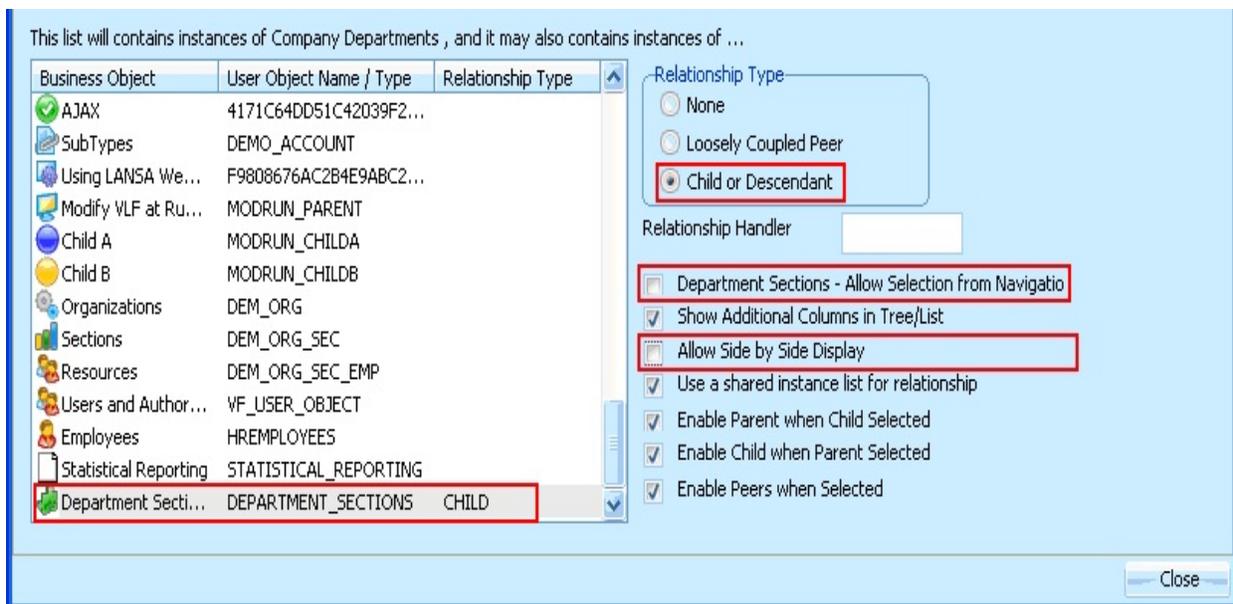
The new business objects are now visible in the iii HR application:



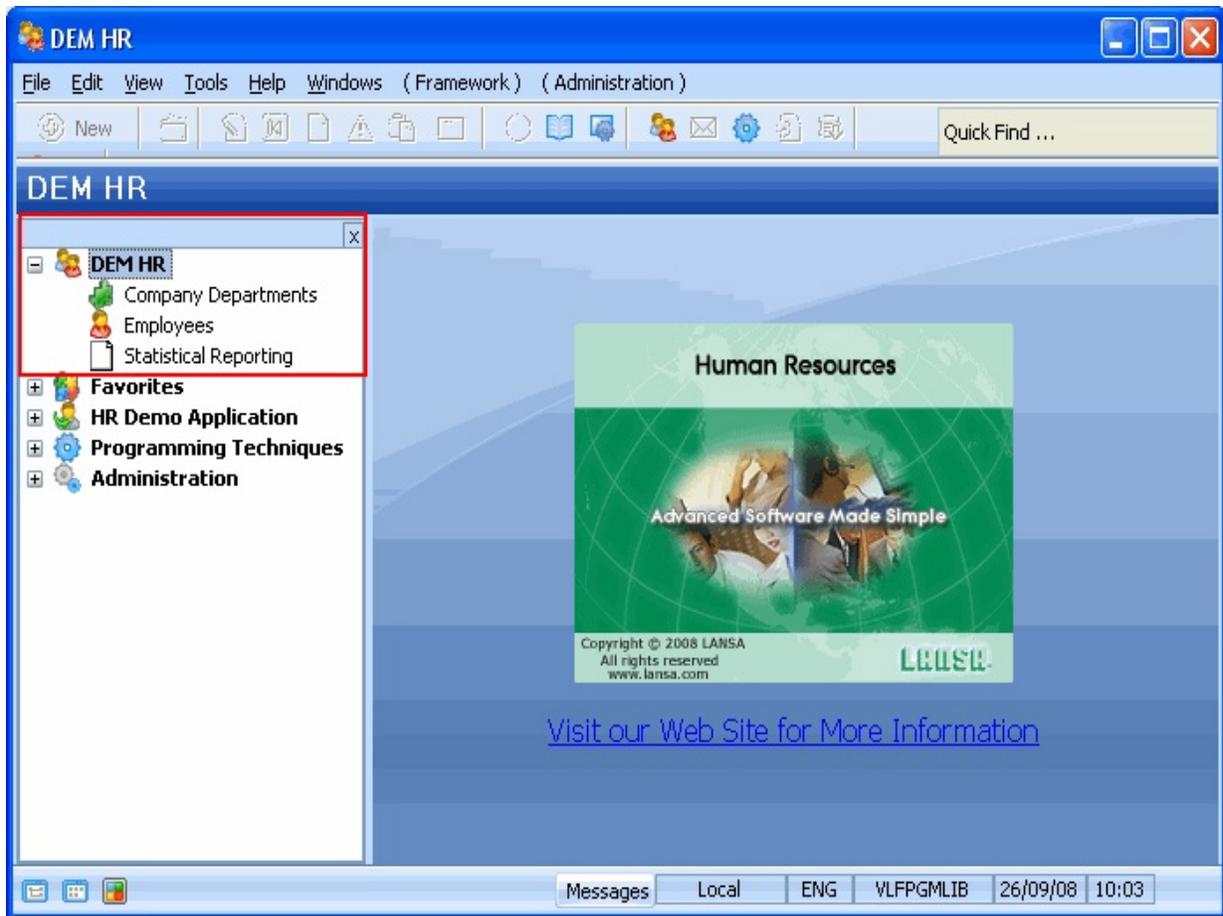
Step 2. Establish the Parent-Child Relationship

In this step you will establish the relationship between the Company Departments and Department Sections business objects.

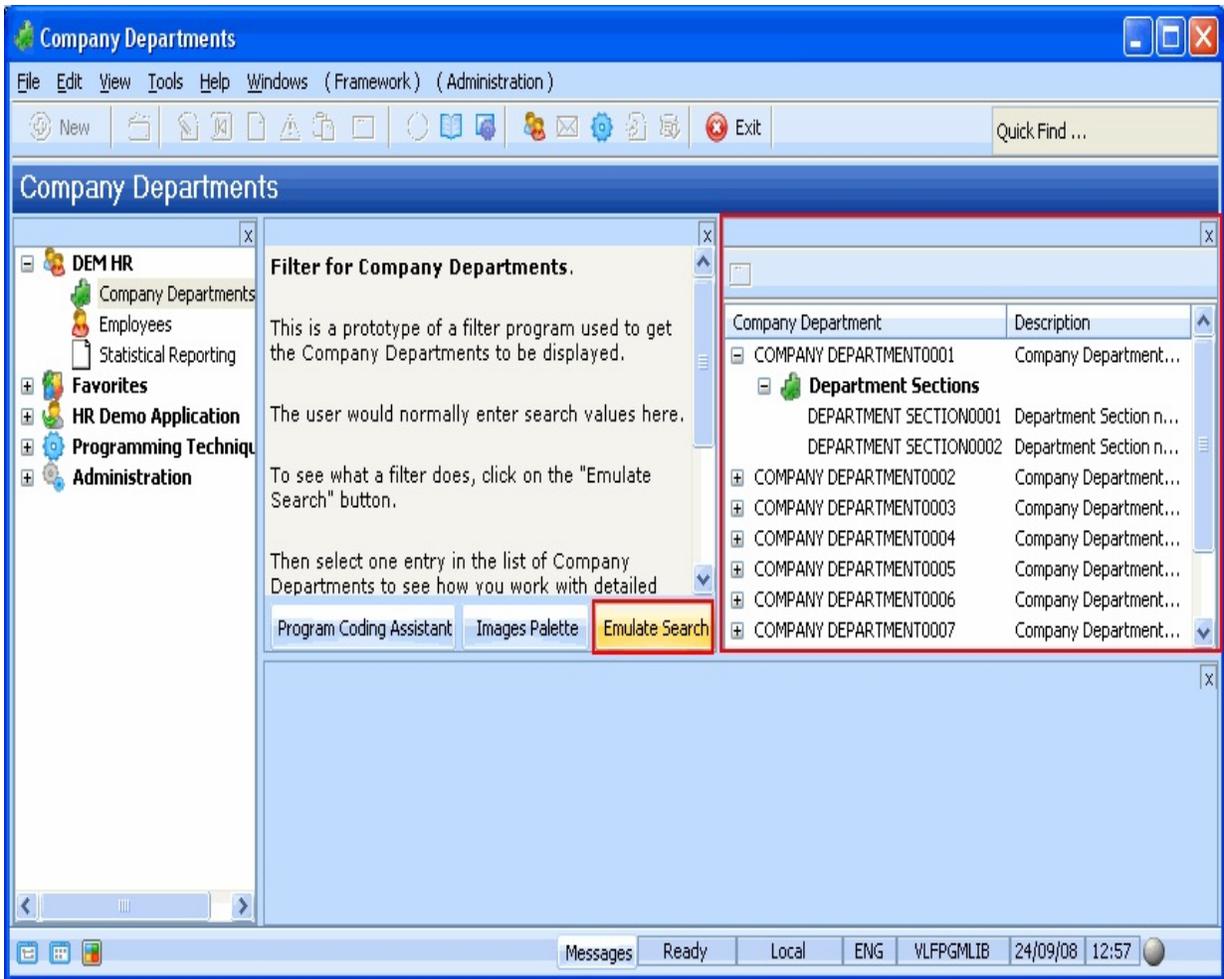
1. Display the properties of the Company Departments object.
2. Click on the Instance List / Relations tab.
3. In the list in the lower left portion of the screen scroll down and select the Department Sections business object.
4. In the Relationship Type group box on the right select the Child or Descendant radio button.
5. Deselect the check box Allow Selection from Navigation Pane.
6. Deselect the check box Allow Side by Side display.
7. Click the Close button in the message that appears.



8. Click the Close button.
9. Save and restart the Framework.
10. Open the iii HR application. Notice that the Department Sections business object is no longer displayed in the navigation pane.



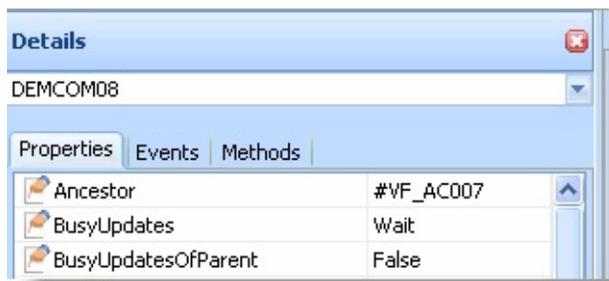
11. Select the Company Departments business object and click on the Emulate Search button in the mock-up filter. Expand one of the Company Departments in the instance list. Notice that the emulated data shows its child business objects, the Department Sections.



Step 3. Create a Hidden Filter for Company Departments

In this step you will create a hidden filter that loads the Company Departments to the instance list when you select the Company Departments business object in the navigation pane. With a hidden filter there is no end-user interaction and the filter is not visible.

1. Display the Visual LANSA Editor.
2. Create a reusable part. Specify iiiCOM08 as the name of your filter and Departments Hidden Filter as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
3. In the Details tab specify VF_AC007 as the Ancestor of the reusable part.



4. Display the Source tab.
5. Create a uInitialize routine after the BEGIN_COM statement:

```
Mthroutine Name(uInitialize) Options(*Redefine)
Endroutine
```

The Endroutine statement may be created automatically if you have the AutoComplete Prompter turned on.

6. In the uInitialize routine make the filter hidden so that all that will show at run-time is the instance list:

```
Set #Com_Owner avHiddenFilter(TRUE)
```

7. Then indicate that the instance list updating is about to start and clear the instance list:

```
Invoke #avListManager.BeginListUpdate  
Invoke #avListManager.ClearList
```

8. Read all the departments and add them to the instance list:

```
Select Fields(#Department #DeptDesc) From_File(DEPTAB)  
  
Invoke #avListManager.AddtoList Visualid1(#DeptDesc)  
Visualid2(#Department) AKey1(#Department)  
BusinessObjectType(COMPANY_DEPARTMENTS) NColumn1(0)  
Acolumn1("") Acolumn2("") Acolumn3("") Acolumn4("") Acolumn5("")  
Acolumn6("") Acolumn7("")  
  
EndSelect
```

Note: It is necessary to initialize the additional columns with the AddToList, as you will later populate the list with entries for Department Sections, which will fill these columns.

9. Lastly indicate that instance list updating is now complete:

```
Invoke #avListManager.EndListUpdate
```

Your code will look like this:

```

Function Options(*DIRECT)
-BEGIN_COM ROLE(*EXTENDS #VF_AC007) HEIGHT(182) WIDTH(326)
  -Mthroutine Name(uInitialize) Options(*Redefine)
    Set #Com_Owner avHiddenFilter(TRUE)

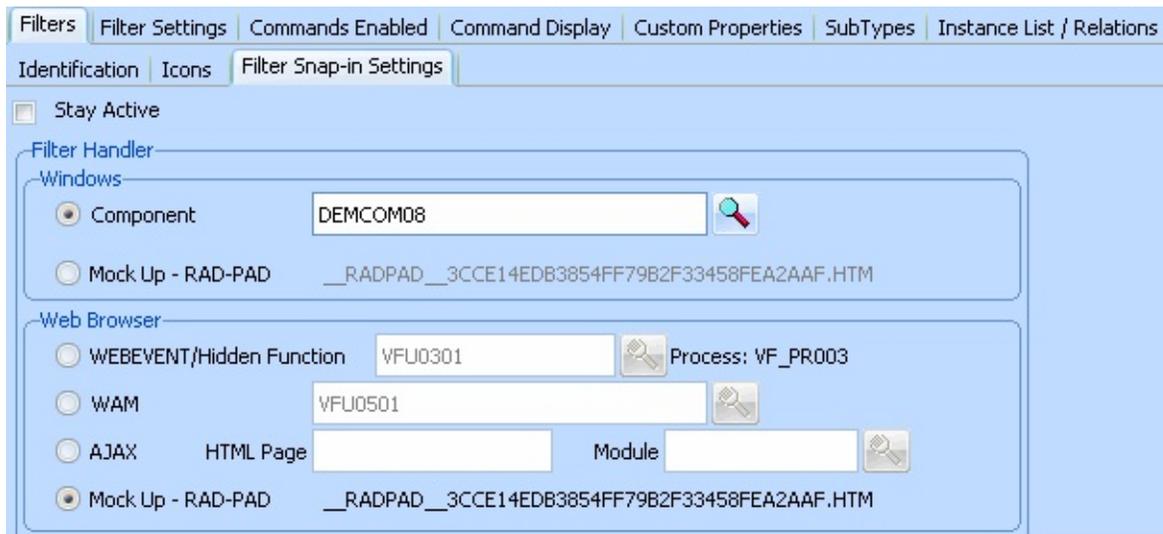
    Invoke #avListManager.BeginListUpdate
    Invoke #avListManager.ClearList

    -Select Fields(#Deptment #DeptDesc) From_File(DEPTAB)
      Invoke #avListManager.AddtoList Visualid1(#DeptDesc) Visualid2(#Deptment)
      AKey1(#Deptment) BusinessObjectType(COMPANY_DEPARTMENTS) NColumn1(0)
      Acolumn1('') Acolumn2('') Acolumn3('') Acolumn4('') Acolumn5('')
      Acolumn6('') Acolumn7('')
    -EndSelect

    Invoke #avListManager.EndListUpdate
  -Endroutine
-End_Com

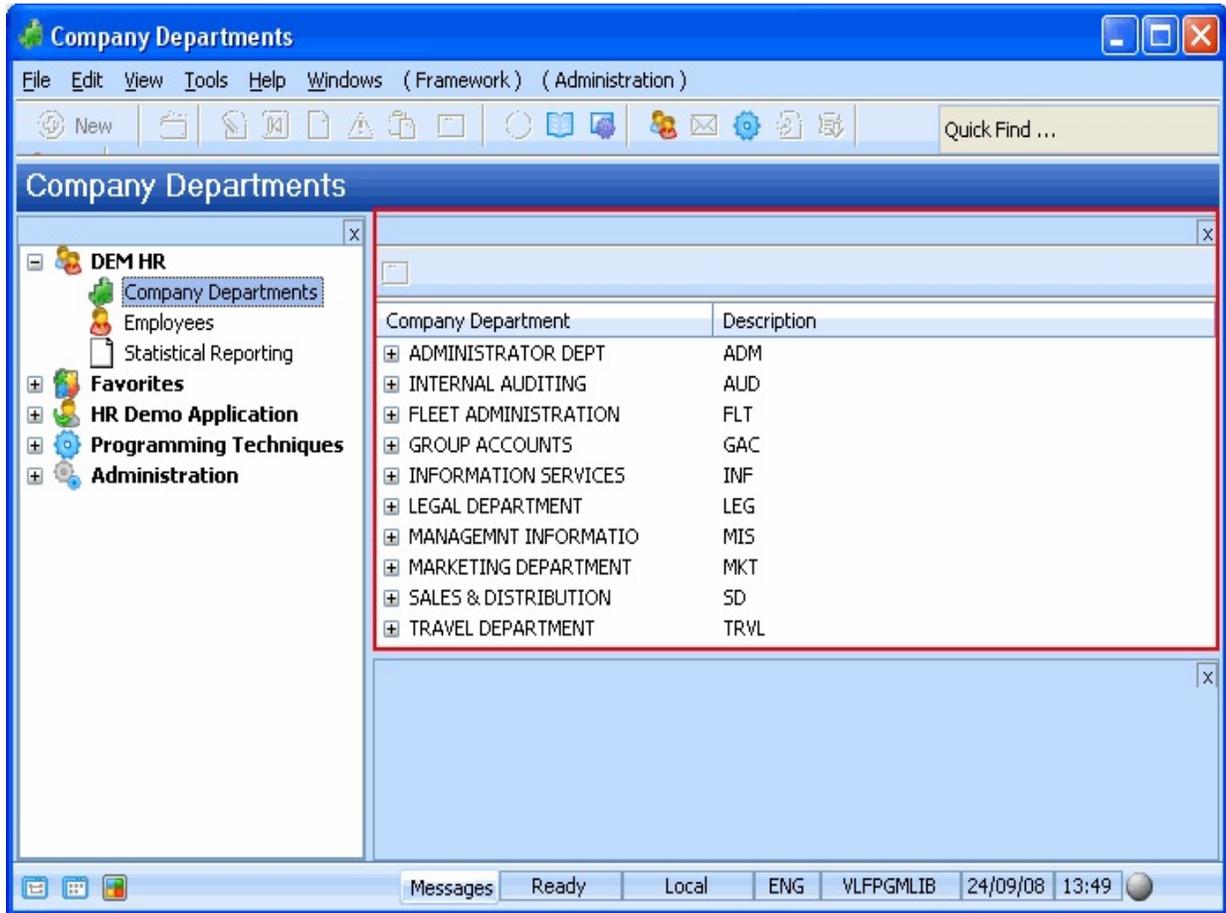
```

10. Compile the filter.
11. Display the Framework and then the properties of the Company Departments object.
12. Display the Filter Snap-in Settings tab.
13. Specify iiiCOM08 as the real filter.



14. Close the Company Departments' properties.
15. Click on Employees and then Company Departments again so that the

hidden filter loads the departments to the instance list.



16. Expand a Department. Notice that no Department Sections are loaded. You will create the relationship handler that loads the sections in the next step.

Step 4. Create a Relationship Handler to Load Sections

In this step you will create a relationship handler that loads Sections into the instance list when a Department is expanded.

You could have loaded all the Sections in the hidden filter code together with the Departments, but by using a relationship handler you can improve filter performance by first only adding root or parent objects to the instance list and then dynamically adding the child objects.

1. In the Visual LANSA editor, create a process iiiPROC2 – Framework Functions. Create a function belonging to this process. Specify iiiFN04 as the name of your function and Relationship Handler as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
2. Replace the existing code in the function with this code that indicates that this function is a relationship handler:

```
FUNCTION OPTIONS(*DIRECT *LIGHTUSAGE) RCV_LIST(#VIS_LIST
#PID_LIST #COL1_LIST #COL2_LIST #COL3_LIST #COL4_LIST
#COL5_LIST #COL6_LIST #COL7_LIST #COL8_LIST #COL9_LIST
#COLA_LIST)
```

```
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL1)
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL2)
```

The VFREL1 and VFREL2 functions which you include contain the standard definitions for relationship builder functions.

3. Next clear all the keys and additional columns in the instance list:

```
EXECUTE SUBROUTINE(CLEARKEYS)
EXECUTE SUBROUTINE(CLEARCOLS)
```

The subroutines you call in the relationship handler are contained in the VFREL2 function.

4. Get the key value of the selected department:

```
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)
```

5. Select the sections in the current department and set the values of the instance list entry:

```
SELECT FIELDS(*ALL) FROM_FILE(SECTAB)
WITH_KEY(#DEPARTMENT)
  EXECUTE SUBROUTINE(SETAKEY) WITH_PARM(1 #DEPARTMENT)
  EXECUTE SUBROUTINE(SETAKEY) WITH_PARM(2 #SECTION)
  EXECUTE SUBROUTINE(SETNCOL) WITH_PARM(1 #SECPCODE)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(1 #SECADDR1)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(2 #SECADDR2)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(3 #SECADDR3)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(4 #SECPHBUS)
  EXECUTE SUBROUTINE(ADDTOLIST)
WITH_PARM('DEPARTMENT_SECTIONS' #SECDESC #SECTION)
ENDSELECT
```

- The SETAKEY subroutine sets the key values of the child instance list. The first parameter of the subroutine is the key position and the second parameter is the value of the key. There is also a SETNKEY subroutine to set a numeric key.
- The SETNCOL and SETACOL subroutines add additional columns for the child instance list entry.
- The ADDTOLIST subroutine adds the entry to the instance list. The first parameter of the subroutine is the child business object name, the second parameter is the Visual ID 1 column and the third parameter is the Visual ID 2 column.

Your code will now look like this:

```

FUNCTION OPTIONS(*DIRECT *LIGHTUSAGE)
  RCV_LIST(#VIS_LIST #PID_LIST #COL1_LIST #COL2_LIST #COL3_LIST #COL4_LIST #COL5_LIST
    #COL6_LIST #COL7_LIST #COL8_LIST #COL9_LIST #COLA_LIST)

⊕ INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL1)
⊕ INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL2)

* Clear all keys and additional columns
EXECUTE SUBROUTINE(CLEARKEYS)
EXECUTE SUBROUTINE(CLEARCOLS)

* Return the sections in a department/organization
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)

* Read all the Sections in the specified
* department and add them to the instance list

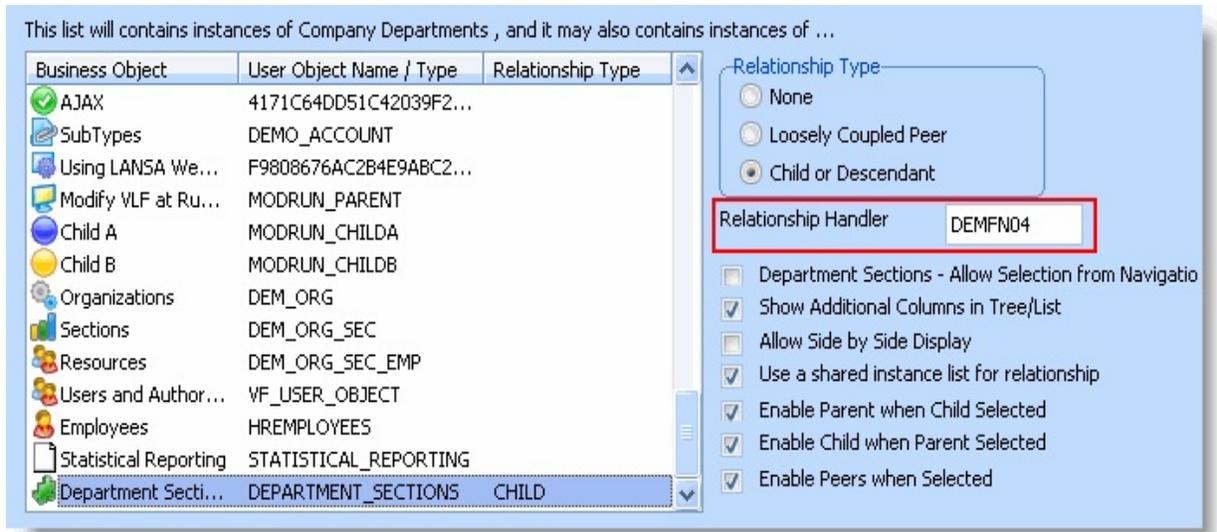
⊖ SELECT FIELDS(*ALL) FROM_FILE(SECTAB) WITH_KEY(#DEPARTMENT)
DCM0123/Warning : no key specified to match file SECTAB from DC@DEMOLIB key field SECTION

EXECUTE SUBROUTINE(SETAKEY) WITH_PARMS(1 #DEPARTMENT)
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMS(2 #SECTION)
EXECUTE SUBROUTINE(SETNCOL) WITH_PARMS(1 #SECPCODE)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(1 #SECADDR1)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(2 #SECADDR2)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(3 #SECADDR3)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(4 #SECPHBUS)
EXECUTE SUBROUTINE(ADDTOLIST) WITH_PARMS('DEPARTMENT_SECTIONS' #SECDESC #SECTION)

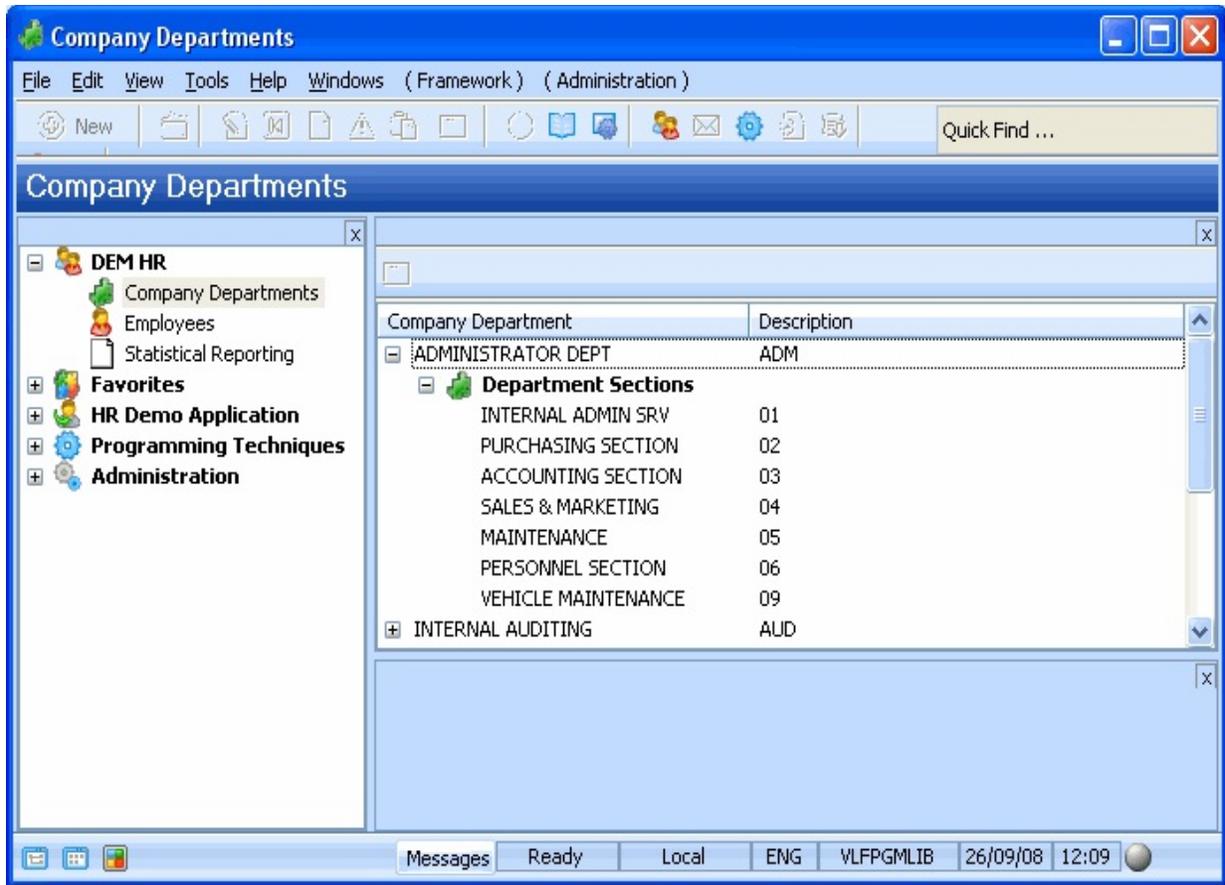
ENDSELECT

```

6. Compile the function.
7. Display the Framework.
8. Display the properties of the Company Departments business object.
9. In the Instance List/Relations tab select the Department Sections business object.
10. In the Relationship Handler field, type in the name of the relationship handler.



12. Close the Company Departments properties.
13. Save and restart the Framework.
14. Select the Company Departments business object in the iii HR application.
15. Expand a department in the instance list.



The sections in each department you expand are loaded dynamically.

Note that only the section name and identifier are shown in the list. In the next step you change the instance list to show additional columns for the sections.

Step 5. Display Additional Columns in the Instance List

In this step you create additional columns in the instance list to show all the data loaded for sections by the relationship handler.

1. Display the properties of the Company Departments business object.
2. In the Instance List / Relationships tab specify these additional columns:

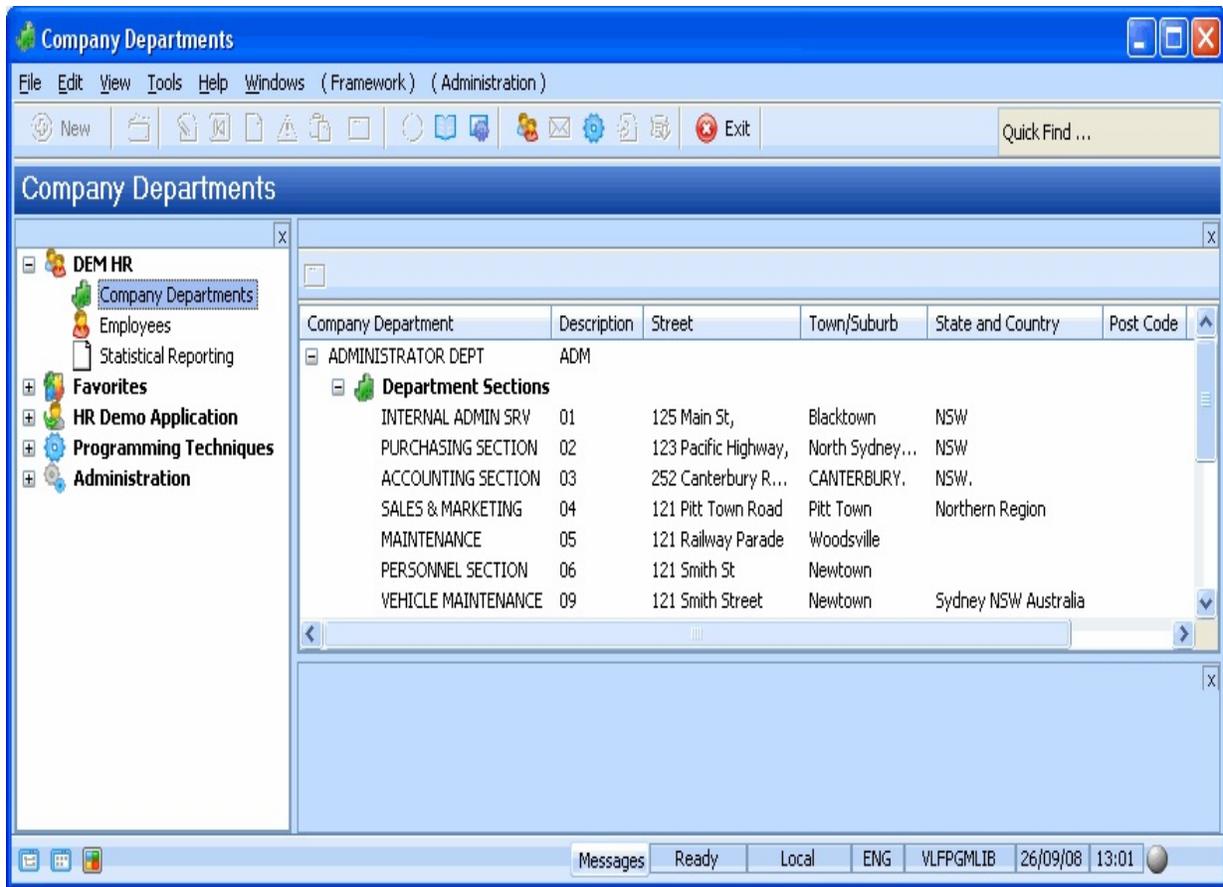
Column Sequence Column Type Column Caption

30	ACOLUMN1	Street
40	ACOLUMN2	Town/Suburb
50	ACOLUMN3	State and Country
60	NCOLUMN1	Post Code
70	ACOLUMN4	Phone

Your instance list column definitions now look like this:

Sequence	Type	Caption	Width % (Total 25%)
10	VISUALID1	Company Department	25
20	VISUALID2	Description	
30	ACOLUMN1	Street	
40	ACOLUMN2	Town/Suburb	
50	ACOLUMN3	State and Country	
60	NCOLUMN1	Post Code	
70	ACOLUMN4	Phone	
	ACOLUMN5		

3. Close the properties of the Company Departments object. Your instance list now shows the additional columns for the sections:

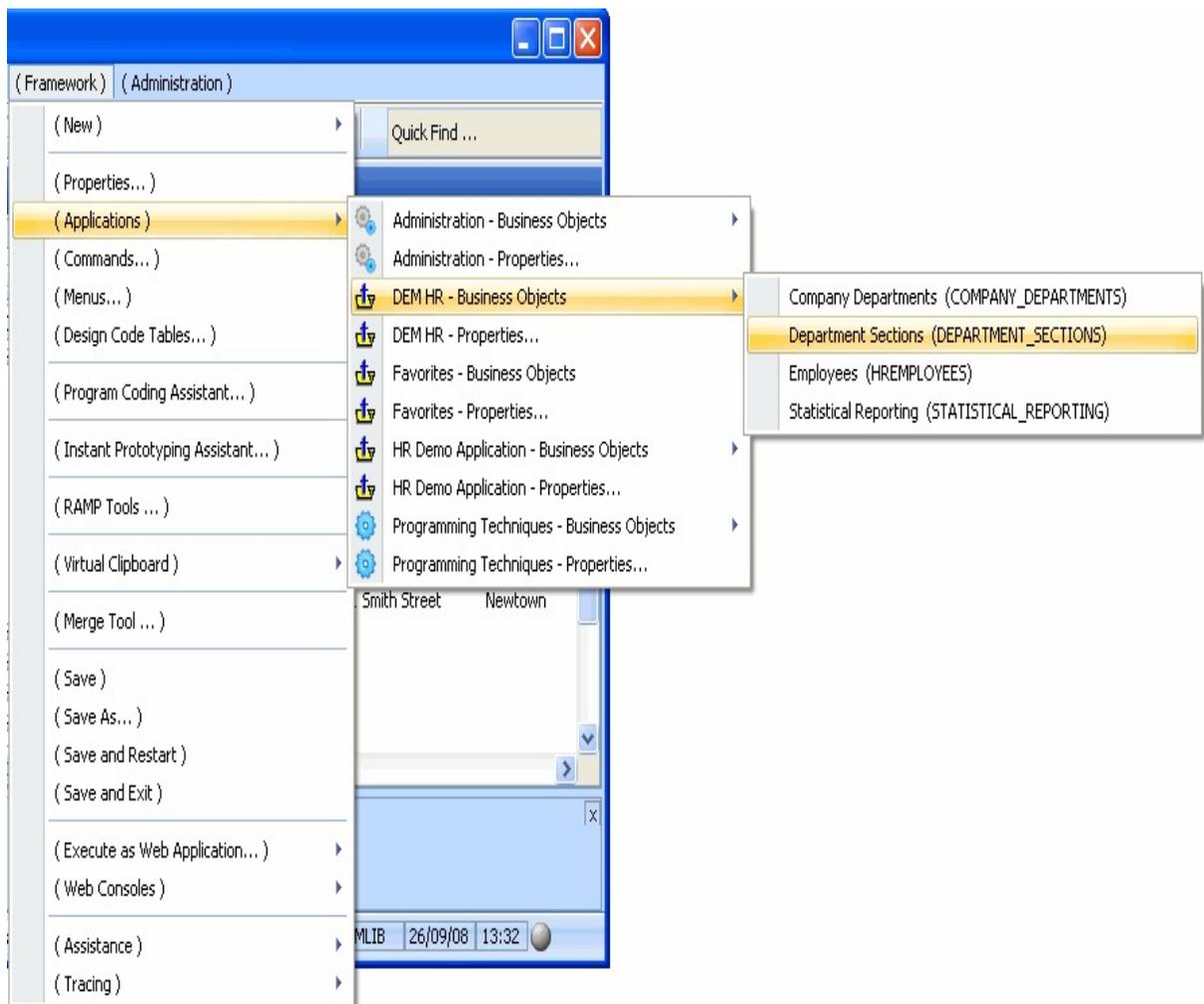


3. Close the properties window and save the Framework.
4. Select the iii HR application in the web Framework and then the Company Departments business object
5. Expand a department in the instance list and then the sections underneath it.

Step 6. Access the Properties of Hidden Child Objects

In this step you will learn how to access the properties of the hidden child business object Department Sections which is not visible in the navigation pane.

1. Display the Framework menu and select the Applications... menu option.
2. Select the iii HR application.
3. Select the Department Sections business object to display the properties of the Department Sections business object.



4. Close the properties of the Department Sections business object.

There is also an alternative way of displaying the properties of child business

objects which are not accessible from the navigation pane:

5. Display the sections in a department in the instance list.
6. Double-click on a section to display the properties of the Department Sections business object.

Summary

Important Observations

- You can create instance lists that contain more than one type of object. You do this by defining relationships between business objects. The relationships can either be peer-to-peer or parent-child.
- In situations where you want to completely fill the business object instance list programmatically, the filter has no meaningful interaction with the end-user and can be hidden from view.
- A relationship handler is an RDML function that is called to dynamically expand the relationship between a parent and child object. By doing this you can improve filter performance by only adding root or parent objects to the instance list initially.
- The Framework instance list can display up to 10 alphanumeric and/or 10 numeric additional columns in an instance list.

Tips & Techniques

- The Advanced section of the Programming Techniques sample application has examples of advanced instance lists.
- LANSAs supplies a sample relationship handler to copy from when creating your relationships. The source is stored in function DF_REL01 in the process DF_PROC.

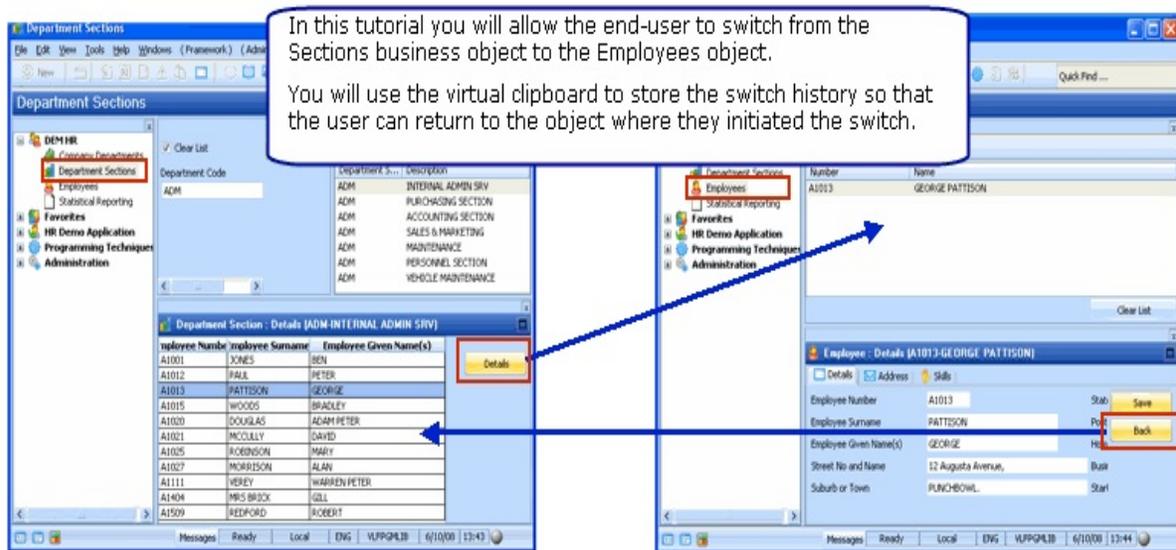
What I Should Know

- How to create a parent-child relationship between business object
- How to create a hidden filter
- How to write a relationship handler
- How to add additional columns to the instance list

VLF012WIN - Controlling Navigation Using Switching and the Virtual Clipboard

Objectives

- To learn how to use switching to swap control between different business objects and to execute commands at the Framework, application or business object level (see [Object Switching Service](#)).
- To learn to use [The Virtual Clipboard](#) to store the switch history.



To achieve this objective, you will complete the following steps:

Step 1. [Create a Filter for Department Sections](#)

Step 2. [Create a Details Command Handler for Department Sections](#)

Step 3. [Add Logic to Switch from Sections to the Employees Business Object](#)

Step 4. [Record Switch History using the Virtual Clipboard](#)

Step 5. [Use the Switch History to Return to the Original Business Object](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

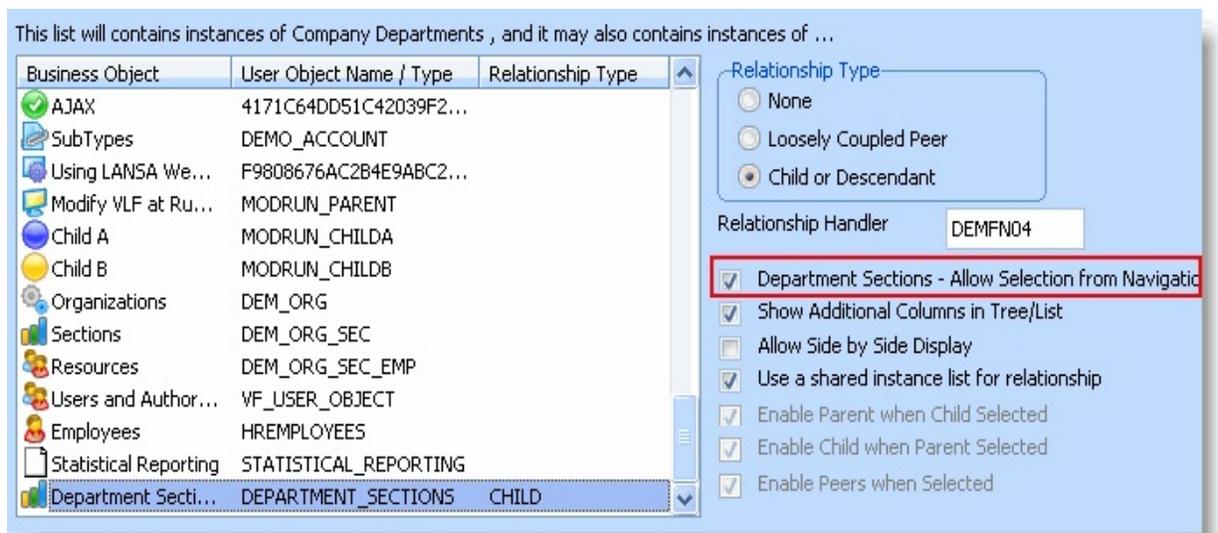
- Tutorials VLF000 – VLF007WIN and VLF011WIN

Step 1. Create a Filter for Department Sections

In this step you will make the Department Sections business object visible in the navigation pane and create a filter for it.

You need to do this in preparation for the switching exercise because object switching can only be performed on objects which are visible in the navigation pane.

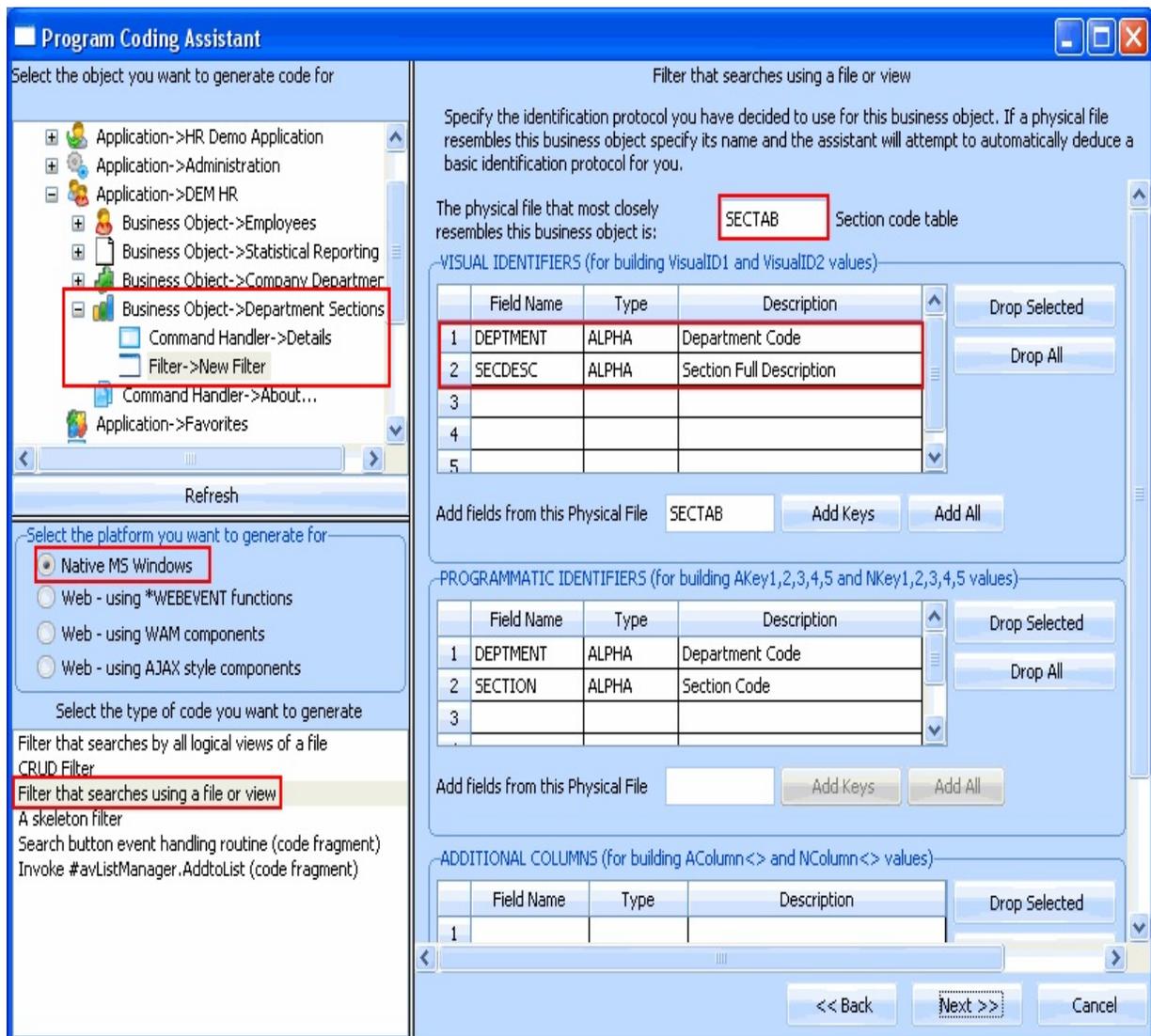
1. In the Framework, display the properties of the Company Departments business object.
2. Display the Instance List / Relationships tab.
3. Select Department Sections in the list on the bottom left.
4. Select the option Department Sections – Allow Selection from Navigation Pane.



5. Close the properties of the Company Departments business object. The Department Sections business object is now visible in the navigation pane.
6. Display the properties of the Department Sections business object.
7. Change the icon for example to 🏢.

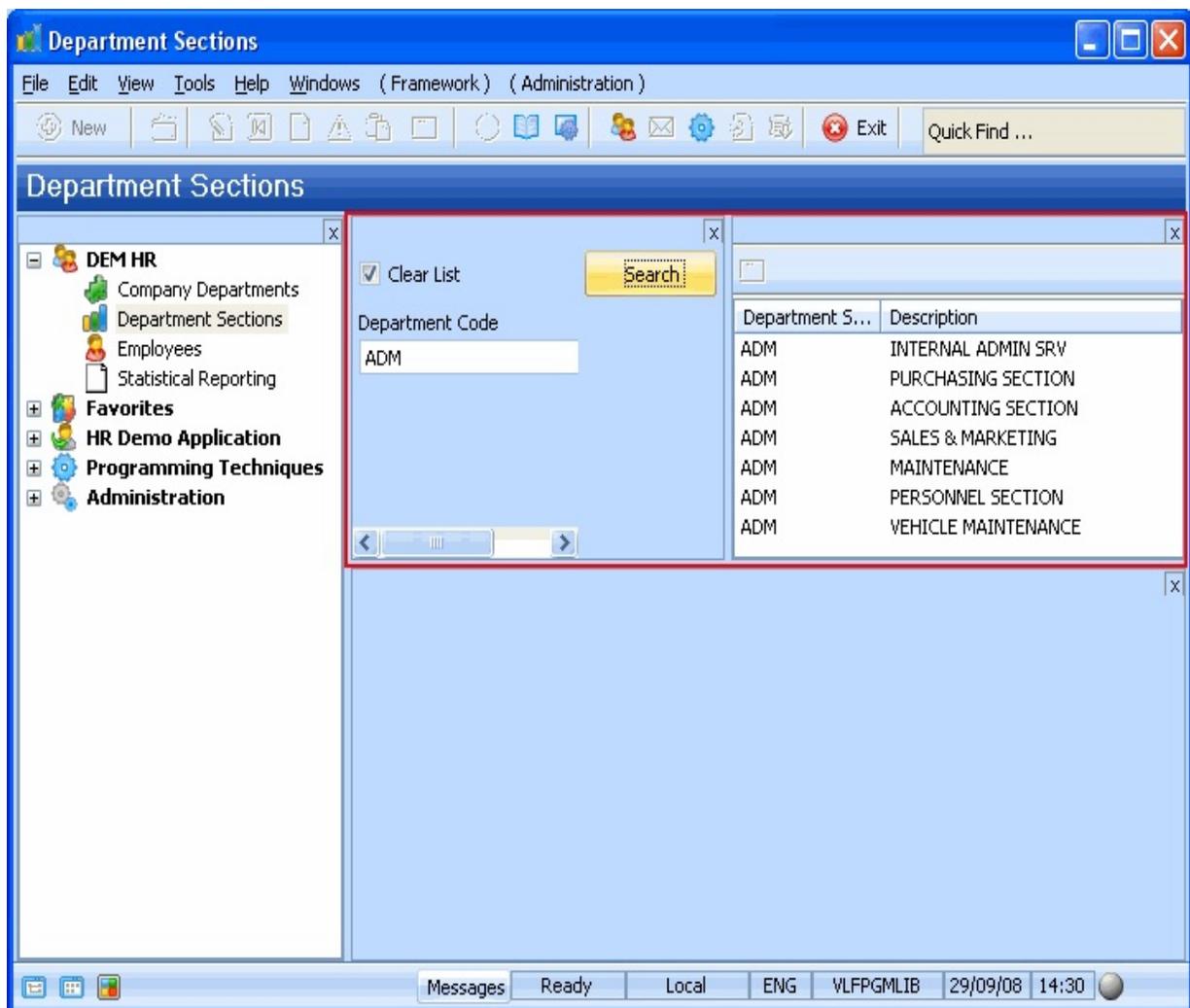
Next you need to replace the mock-up filter in the Sections business object with a functional filter to populate the instance list:

6. Start the Program Coding Assistant.
7. Select the Department Sections business object in the iii HR application.
8. Select New Filter, Windows as the platform and a Filter that searches a file or a view.
9. Click Next.
10. Specify SECTAB as the physical file, and DEPARTMENT and SECDESC as the visual identifiers.



11. Accept the other defaults set by the Program Coding Assistant and click Next.

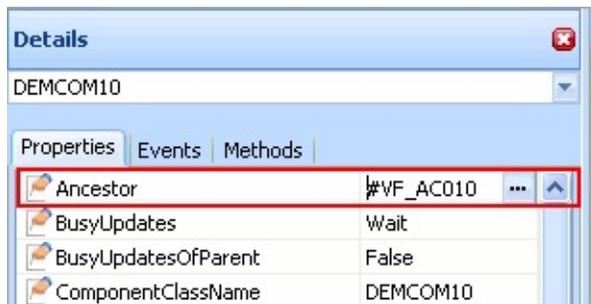
12. Specify DEPARTMENT field as the key to be used for search operations.
13. Click Next.
14. Click Generate Code.
15. On the Generated Code page specify iiiCOM09 as the name of your filter and Sections Filter as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
16. Click Create. The component is displayed in the Visual LANSA Editor.
17. Compile the filter.
18. In the Framework, snap the filter in the Department Sections business object.
19. Test the filter.



Step 2. Create a Details Command Handler for Department Sections

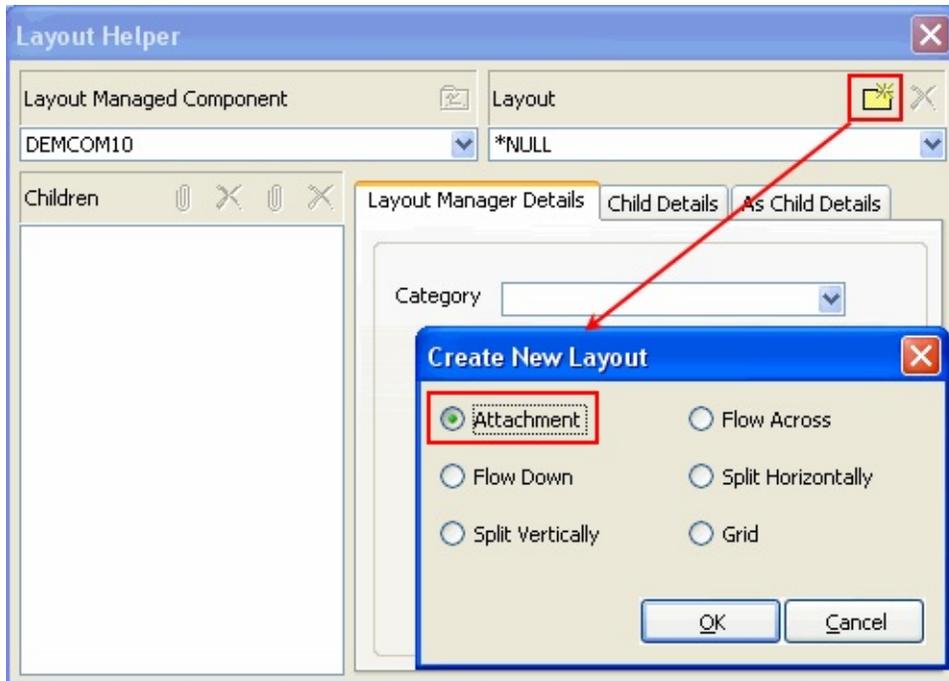
In this step you will create a Details command handler which will show the employees in the selected Section.

1. Create a reusable part. Specify iiiCOM10 as the name of your command handler and Section Details as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
2. Click on the Details tab and specify VF_AC010 as the Ancestor of your component.



You will first add a layout manager to the command handler to control the placement of the controls on it:

3. Start the Layout Helper from the View menu.
4. Add an Attachment Manager to the reusable part iiiCOM10.

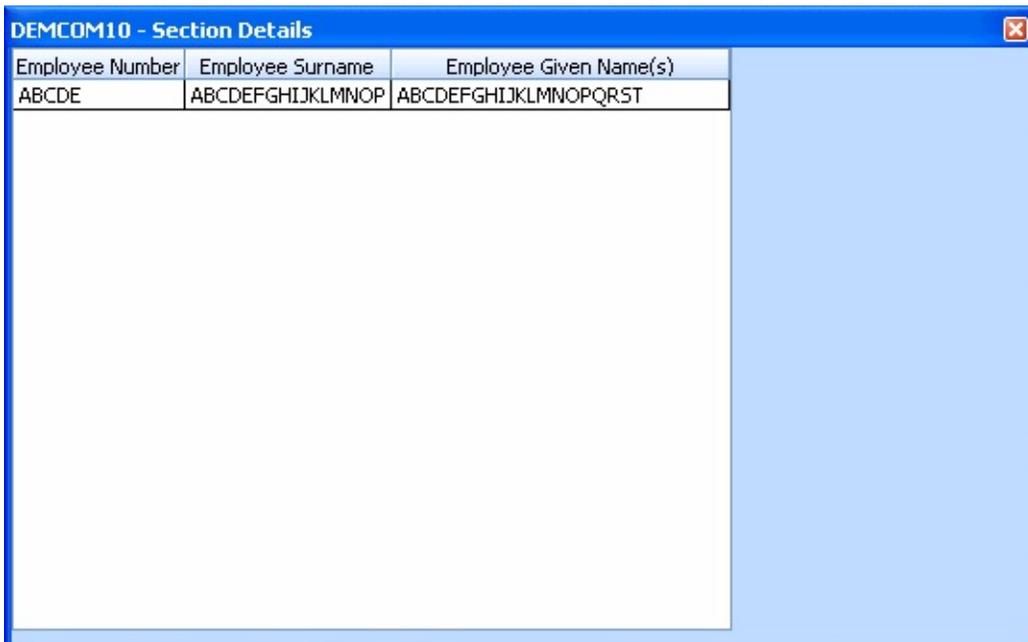


5. Display the Common Controls tab on the Favorites tab.
6. Drag and drop a panel to the right hand side (PANL_1). Change its Width to 94. This area will later contain the Details push button.
7. Drag and drop another Panel over the centre of the remaining area (PANL_2). Use the Layout Helper to add the attachment manager (ATLM_1) to this panel using the dropdown on the right hand side.
8. Drag and drop the grid control to the centre of PANL_2. Your grid will be attached to the sides of PANL_2. You have created a command handler which will automatically resize with the rest of the framework.

Your command handler will now look like this:



9. Display the PSLMST file in the repository and expand it.
10. Drag the fields EMPNO, SURNAME and GIVENAME to the grid.
11. Make the WidthType of the GIVENAME column (GDCL_3) Remainder.
12. Make the SelectionStyle of the grid WholeRow.



Now write the code to populate the Employee grid in the command handler:

13. Display the Source tab.

14. Add a uExecute method routine after the BEGIN_COM statement:

```
Mthroutine uExecute Options(*Redefine)
```

The uExecute method is invoked whenever the user executes the Framework command that is associated with the command handler.

15. Use the GetCurrentInstance method to get the current Department and Section.

```
Invoke #avListManager.GetCurrentInstance AKey1(#DEPARTMENT)  
AKey2(#SECTION)
```

16. Clear the Employee grid:

```
Clr_list #Grid_1
```

17. Lastly select the employees that belong to the section from the PSLMST1 logical view and add them to the grid:

```
Select fields(#EMPNO #SURNAME #GIVENAME) from_file(PSLMST1)  
with_Key(#DEPARTMENT #SECTION)
```

```
Add_entry #GRID_1
```

```
Endselect
```

Your code will now look like this:

```

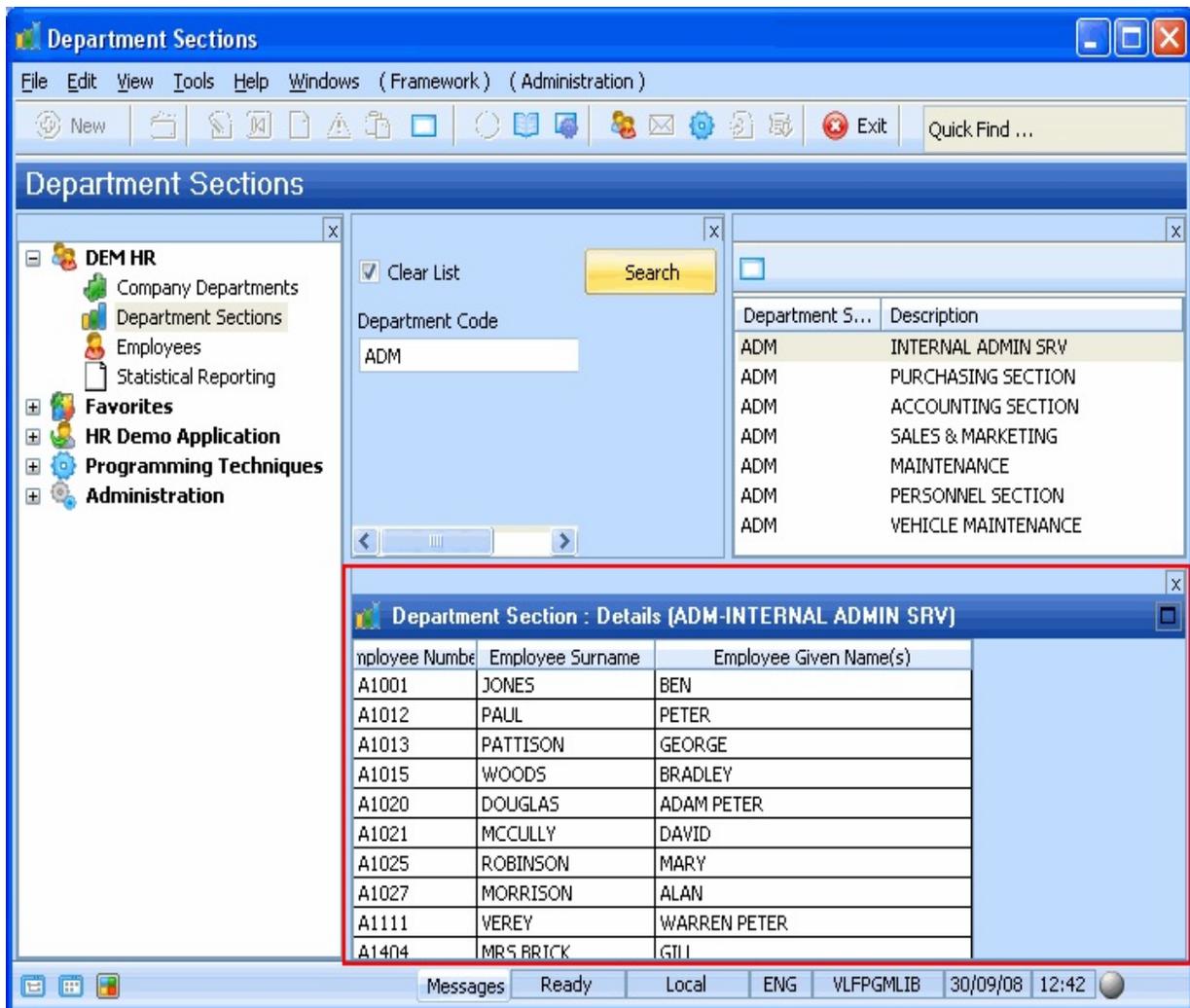
* *****
FUNCTION OPTIONS(*DIRECT)
-BEGIN_COM ROLE(*EXTENDS #VF_AC010) LAYOUTMANAGER(#ATLM_1)
  -DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
    -MTHROUTINE NAME(uExecute) OPTIONS(*REDEFINE)
      Invoke #avListManager.GetCurrentInstance AKey1(#DEPARTMENT) AKey2(#SECTION)

      Clr_list #GRID_1

      -Select fields(#EMPNO #SURNAME #GIVENAME) from_file(PSLMST1) with_Key(#DEPARTMENT #SECTION)
        Add_entry #GRID_1
      -Endselect
    -ENDROUTINE
  -END_COM

```

18. Compile your command handler.
19. Snap the command handler in the Details command of the Department Sections business object.
20. Test your command handler.

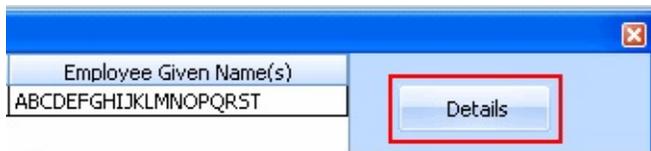


Step 3. Add Logic to Switch from Sections to the Employees Business Object

In this step you will add logic to the Sections' Details command handler to display the details of a selected employee in the Details command handler of the Employees business object.

The switch to the Employees' Details command handler is executed in a button click event.

1. Display the Design tab of the Details command handler.
2. Drag a push button from the Common Controls tab on to the right hand panel (PANL_1) on the command handler.
3. Make the Caption of the button Details.



4. Add a Click event for the button.
5. In the click event add a statement to switch to the Details command handler of the Employees business object.

```
#avframeworkmanager.avSwitch To(BUSINESSOBJECT)  
NAMED(EMPLOYEES) EXECUTE(DETAILS) Caller(#com_owner)  
ClearInstanceList(TRUE)
```

- The To parameter contains BUSINESSOBJECT to indicate the switch is to a business object (you can also switch to the Framework or an application).
- The NAMED parameter must contain your actual business object name.
- The EXECUTE parameter contains the name of the command to execute.
- You can optionally clear the instance list by specifying the ClearInstanceList parameter.

6. Next add the following event routine which will tell the Employees business object which instance should be displayed based on the value of the employee in the grid:

```
Evtroutine Handling(#avFrameworkManager.avAddSwitchInstances)
Caller(#Caller) Options(*NOCLEARERRORS *NOCLEARMESSAGES)

* Make sure the caller is this component
If_ref #Caller is_not(*Equal_to #Com_Owner)
Return
Endif

Invoke Method(#avFrameworkManager.avAddSwitchInstance)
BusinessObjectType(EMPLOYEES) Visualid1(#EMPNO)
Visualid2(#SURNAME) Akey1(#EMPNO)

Endroutine
```

- The avAddSwitchInstances event routine is always executed immediately after you execute a switch using the avSwitch method. This event allows you to control what data will be placed in the instance list of the target business object. The component signaling this event is passed in the Caller parameter.
- It is important to only execute the code in this event if the component that signaled this event is the component itself. Therefore you should return from this event routine if the caller is not equal to #com_owner. Notice how the is_not(*Equal_to is used to compare the #Caller and #Com_Owner. You must use this syntax due to the fact that you are comparing the component itself and not a simple string.
- The avAddSwitchInstance method specifies what data to add in the target instance list.
- There is no reason that you couldn't call the avAddSwitchInstance method repeatedly to place multiple entries into the target business object's instance list.

Your code should now look like this:

```

EvtRoutine Handling(#avFrameworkManager.avAddSwitchInstances) Caller(#Caller)
  Options(*NOCLEARERRORS *NOCLEARMESSAGES)

  * Make sure the caller is this component
  If_ref #Caller is_not(*Equal_to #Com_Owner)
    Return
  Endif

  Invoke Method(#avFrameworkManager.avAddSwitchInstance) BusinessObjectType(EMPLOYEES)
    Visualid1(#EMPNO) Visualid2(#SURNAME) Akey1(#EMPNO)
Endroutine

```

7. Compile the command handler.
8. Test the switching: when you select an employee and click on the Details button on the Sections' Details Command Handler, the Employees business object should be displayed with the selected employee details.

Department Sections

File Edit View Tools Help Windows (Framework) (Administration)

Quick Find ...

Department Sections

DEM HR

- Company Departments
- Department Sections
- Employees
- Statistical Reporting
- Favorites
- HR Demo Application
- Programming Techniques
- Administration

Clear List Search

Department Code: ADM

Department S...	Description
ADM	INTERNAL ADMIN SRV
ADM	PURCHASING SECTION
ADM	ACCOUNTING SECTION
ADM	SALES & MARKETING
ADM	MAINTENANCE
ADM	PERSONNEL SECTION
ADM	VEHICLE MAINTENANCE

Department Section : Details [ADM-INTERNAL ADMIN SRV]

Employee Number	Employee Surname	Employee Given Name(s)
A1001	JONES	BEN
A1012	PAUL	PETER
A1013	PATTISON	GEORGE
A1015	WOODS	BRADLEY
A1020	DOUGLAS	ADAM PETER
A1021	MCCULLY	DAVID
A1025	ROBINSON	MARY
A1027	MORRISON	ALAN
A1111	VEREY	WARREN PETER
A1404	MRS BRICK	GILL
A1509	REDFORD	ROBERT

Messages Ready Local ENG VJPPGMLB

Employees

File Edit View Tools Help Windows (Framework) (Administration)

Quick Find ...

Clear List Employee Surname

DEM HR

- Company Departments
- Department Sections
- Employees
- Statistical Reporting
- Favorites
- HR Demo Application
- Programming Techniques
- Administration

Number	Name
A1013	GEORGE PATTISON

Clear List

Employee : Details (A1013-GEORGE PATTISON)

Details Address Skills

Employee Number: A1013 Home Phone No: [] Save

Employee Surname: PATTISON Business Phone: []

Employee Given Name(s): GEORGE Start date (Y/M): []

Street No and Name: 12 Augusta Avenue, Termination Da: []

Suburb or Town: PUNCHBOWL Department Co: []

State and Country: NSW Section Code: []

Post / Zip Code: 2016 Employee Salar: []

Messages Ready Local ENG VJPPGMLB 6/10/06 10:35

Step 4. Record Switch History using the Virtual Clipboard

In this step you will record the switch history using the virtual clipboard so that the end-user will be able return to the object that initiated the switch.

To use the virtual clipboard most effectively you need to devise a standardized naming protocol for items that are posted onto it. In this exercise you will use this standard to store the switch history:

ID1 SWITCH_HISTORY
ID2 Target Business Object Name
ID3 Target Command Name
ID4 OBJECT_NAME or COMMAND_NAME
FromAValue <object or command name>

In effect you will be storing a switch history table on the clipboard. The first key or ID is the code SWITCH_HISTORY to indicate that all records with this ID are related to switching history.

The ID2 and ID3 contain which business object and command respectively that you are switching to. ID4 contains where you came from. Therefore you need to add two records to the virtual clipboard; one where ID4 equals OBJECT_NAME (the business object) and another where ID4 equals COMMAND_NAME (the command).

1. Display the Source tab of the Sections' Details command handler.
2. In the PHBN_1.Click event, before the avSwitch command, write this code to add the appropriate records to the switch history:

```
* Save to clipboard return list
#avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY)
WithID2(EMPLOYEES) WithID3(DETAILS) WithID4(OBJECT_NAME)
FromAValue(#com_owner.Avobjecttype)
```

```
#avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY)
WithID2(EMPLOYEES) WithID3(DETAILS)
WithID4(COMMAND_NAME) FromAValue(#com_owner.avcommandtype)
```

Use your business object name for the WithID2() parameter.

Note that the actual business object name and command name are placed in the clipboard using the FromAValue parameter. Notice how you can use avobjecttype to get the current business object name and avcommandtype to get the current command name. You should not hard code these values.

Your code should now look like this:

```
EVROUTINE HANDLING(#PHEN_1.Click)
  * Save to clipboard return list
  #avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY) WithID2(EMPLOYEES) WithID3(DETAILS)
    WithID4(OBJECT_NAME) FromAValue(#com_owner.Aobjecttype)
  #avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY) WithID2(EMPLOYEES) WithID3(DETAILS)
    WithID4(COMMAND_NAME) FromAValue(#com_owner.avcommandtype)

  #avframeworkmanager.avSwitch To(BUSINESSOBJECT) NAMED(EMPLOYEES) EXECUTE(DETAILS)
    Caller(#com_owner) ClearInstanceList(TRUE)
ENDROUTINE
```

3. Compile the command handler.
4. Close the command handler.

Step 5. Use the Switch History to Return to the Original Business Object

In this step you will use the switch history to allow the end-user to return to the Sections business object from where they initiated the switch.

1. Open the Employees' Details command handler iiiCOM03.
2. Display the Common Controls tab in the Favorites tab and drag a push button under the Save button.
3. Make the caption of the button Back.
4. Make the name of the button BACK_BTN.
5. Add a Click event for the button.
6. In the Click event add this code so that when the users click on the button, they will be switched back to the business object from which they came:

```
EVTROUTINE HANDLING(#BACK_BTN.Click)

define field(#ff_objnme) TYPE(*CHAR) LENGTH(32) DESC('Object
Name')
define field(#ff_cmdnme) TYPE(*CHAR) LENGTH(32) DESC('Command
Name')

* Determine the business object name to switch to
#avFrameworkManager.avrestorevalue WithID1(SWITCH_HISTORY)
WithID2(#com_owner.Avobjecttype)
WithID3(#com_owner.Avcommandtype) WithID4(OBJECT_NAME)
ToAValue(#ff_objnme)

* Determine which command within the business object to switch to
#avFrameworkManager.avrestorevalue WithID1(SWITCH_HISTORY)
WithID2(#com_owner.Avobjecttype)
WithID3(#com_owner.Avcommandtype) WithID4(COMMAND_NAME)
ToAValue(#ff_cmdnme)

* Perform the switch
#avframeworkmanager.avSwitch To(BUSINESSOBJECT)
NAMED(#ff_objnme) EXECUTE(#ff_cmdnme) Caller(#com_owner)
```

ENDROUTINE

- When you want to send the user back to the component from which the switch occurred, you need to look at the switch history on the virtual clipboard. Remember that you need to retrieve both the business object and the command to which you need to return. That requires retrieving two values from the virtual clipboard.
- The code first retrieves the OBJECT_NAME or business object value and then the COMMAND_NAME or command value.
- Again, remember that you don't want to hard code the component name, which is why avobjecttype (business object name) and avcommandtype (command name) were used as the values to the WithID2 and WithID3 parameters.
- When you have these two values you can perform another switch to return to the previous component.
- In the code above, the business object was retrieved into the #ff_objnme field and the command was retrieved into the #ff_cmdnme field. Now you simply use the same technique learned earlier to switch to a business object and execute the command.

Your code should look like this:

```
EVTRoutine HANDLING(#BACK_BTN.Click)

    define field(#ff_objnme) TYPE(*CHAR) LENGTH(32) DESC('Object Name')
    define field(#ff_cmdnme) TYPE(*CHAR) LENGTH(32) DESC('Command Name')

    * Determine the business object name to switch to
    #avFrameworkManager.avrestorevalue WithID1(SWITCH_HISTORY) WithID2(#com_owner.Avobjecttype)
        WithID3(#com_owner.Avcommandtype) WithID4(OBJECT_NAME) ToAValue(#ff_objnme)

    * Determine which command within the business object to switch to
    #avFrameworkManager.avrestorevalue WithID1(SWITCH_HISTORY) WithID2(#com_owner.Avobjecttype)
        WithID3(#com_owner.Avcommandtype) WithID4(COMMAND_NAME) ToAValue(#ff_cmdnme)

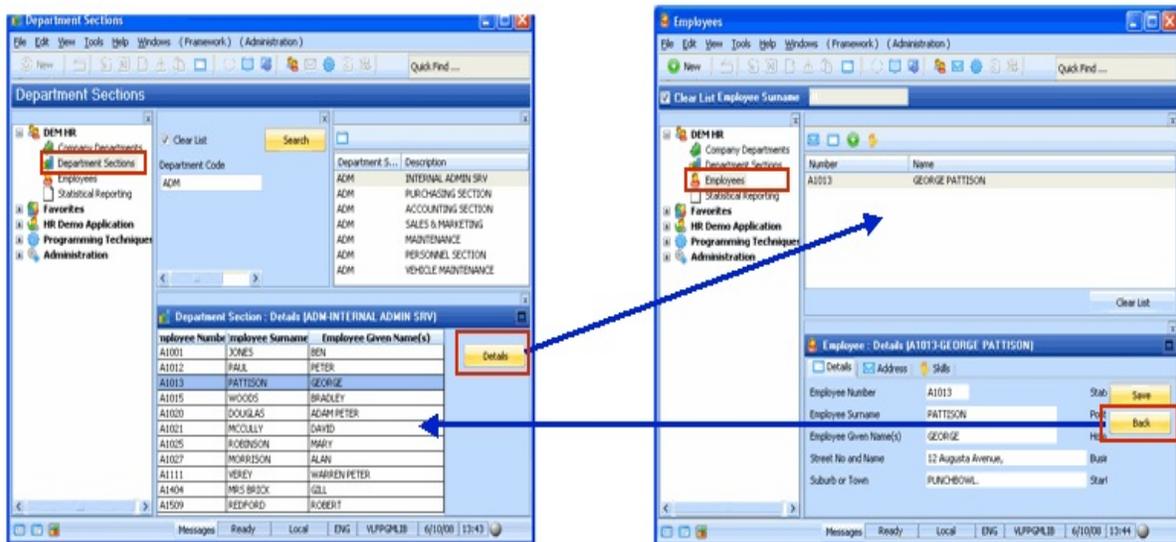
    * Perform the switch
    #avframeworkmanager.avSwitch To(BUSINESSOBJECT) NAMED(#ff_objnme) EXECUTE(#ff_cmdnme)
        Caller(#com_owner)

ENDROUTINE
```

7. Now compile the command handler.

You are now ready to test the switch history:

8. In the Framework select a section from the Department Sections business object.
9. Select an employee from the Sections' Details command handler.
10. Display the details of the selected employee by clicking on the Details button.
11. On the Details command handler of the Employees business object click on the Back button to return to the Sections business object.



Summary

Important Observations

- The Framework switching service allows your filters and command handlers to switch control between different business objects and to execute commands at the Framework, application or business object level.
- The target business object must be able to be selected from the menu (the option Allow selection from the navigation pane in the target business object properties should be checked, and the user should be authorized to the business object), at the time the switch occurs. Switching mimics the actions that a user would perform.
- You can use the Virtual Clipboard for remembering and exchanging information.
- To use the virtual clipboard most effectively you need to devise a standardized naming protocol for items that are posted onto it.

Tips & Techniques

- The Advanced section of the Programming Techniques sample application has examples of switching and remembering values (virtual clipboard).

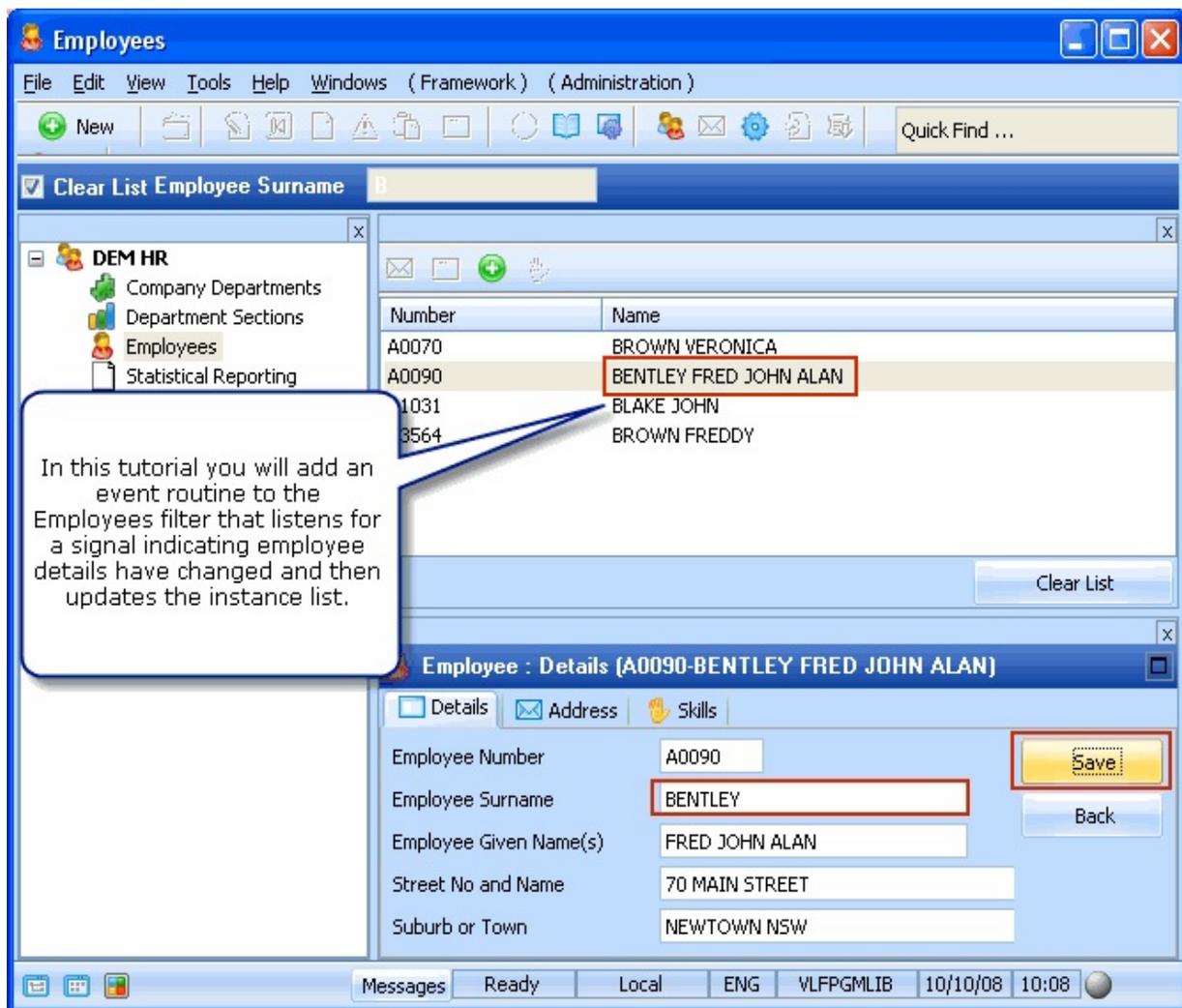
What I Should Know

- How to switch between business objects
- How to use the virtual clipboard to record switch history so that the end-users can switch back to object where the switch was initiated.

VLF013WIN - Signaling Events

Objectives

- To learn how to signal that an event has happened in your filter or command handler to other active filters or command handlers so that they can take appropriate action (see [Event Signaling Service](#)).
- To learn how to update an entry in the instance list (see [Filters and List Manager](#)).



To achieve this objective, you will complete the following steps:

Step 1. [Change Employee Surname and Save the Changes](#)

Step 2. [Add the avSignalEvent to the Employee Details Command Handler](#)

Step 3. [Add a Routine to Listen for the EMPLOYEE_CHANGED Event](#)

Step 4. [Test Signaling](#)

Summary

Before You Begin

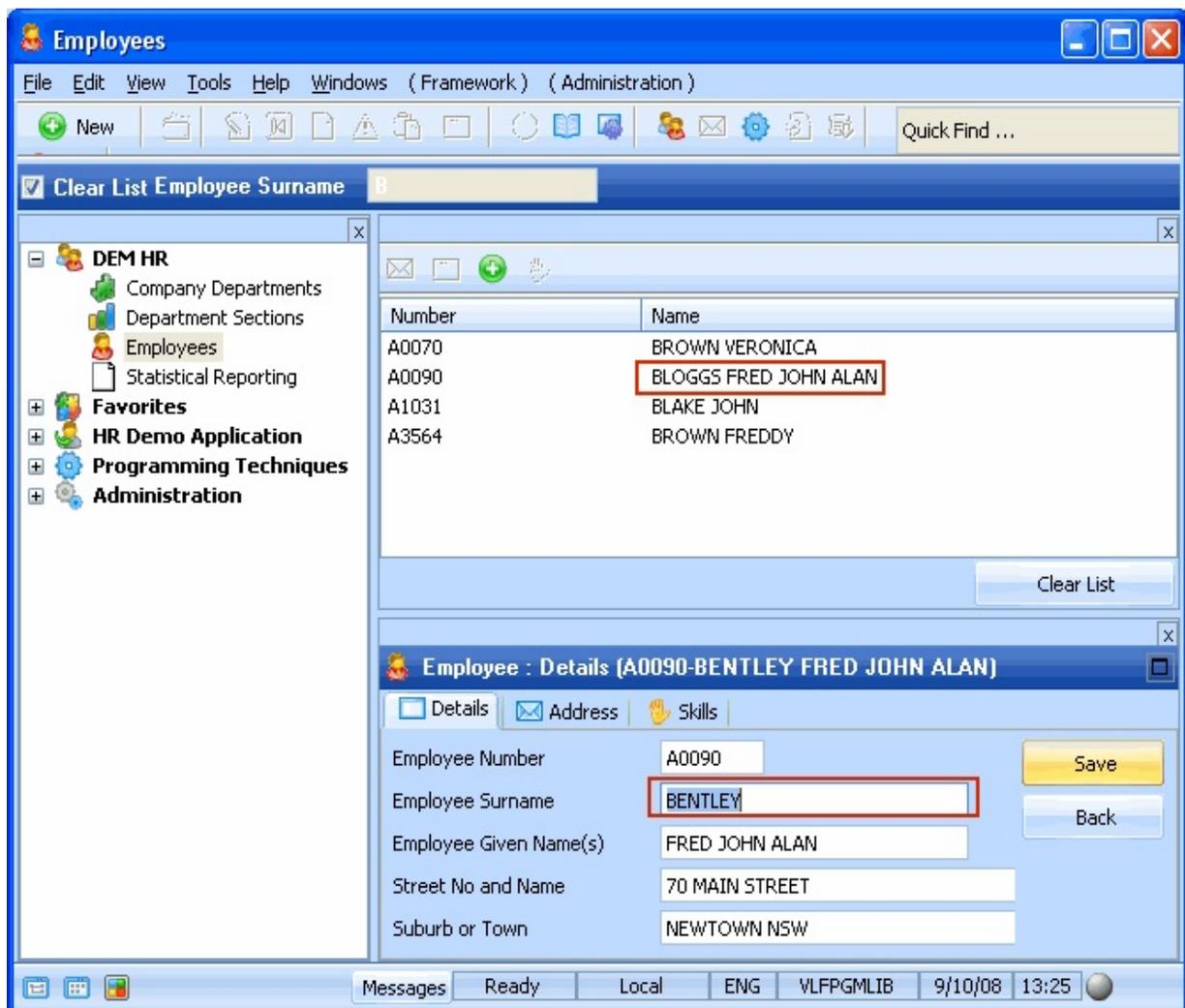
In order to complete this tutorial, you must have completed the following:

- Tutorials VLF000 – VLF007WIN and VLF009WIN

Step 1. Change Employee Surname and Save the Changes

In this step you will change the surname of an employee and save the changes. The instance list will not reflect the change because the filter does not know about the change event.

1. Display the Employees business object in the iii DEM application.
2. Use the mini filter to add entries to the instance list.
3. Select one of the entries and change the employee surname in the Details command handler.
4. Click the Save button. Notice that the new surname is not reflected in the instance list entry:



5. Close the Framework.

In the following steps you will use the signal method in the Details command handler to notify the Employees filter that an employee has changed. You will then add code to update the instance list.

Step 2. Add the avSignalEvent to the Employee Details Command Handler

In this step you change the SAVE_BUTTON event in the Employee Details command handler.

1. Locate and open the Employee Details command handler iiiCOM03.
2. Display the Source tab.
3. Locate the SAVE_BUTTON Click event. Add the following code before the Endroutine:

```
#com_owner.avSignalEvent WITHID(EMPLOYEE_CHANGED)
SENDAINFO1(#EMPNO) TO(FRAMEWORK)
```

```
EvtRoutine Handling(#SAVE_BUTTON.Click) Options(*NOCLEARMESSAGES *NOCLEARERRORS)
    * Check that the connection is still live
    invoke #avFrameworkManager.avCheckConnection ReturnValue(#con_rslt)
    If Cond('*conresult ')
        UPDATE FIELDS(*all) IN_FILE(PSLMST) WITH_KEY(#EMPNO)
        #COM_OWNER.avSignalEvent WithId(EMPLOYEE_CHANGED) Send&Info1(#EMPNO) To(FRAMEWORK)
    endif
Endroutine
```

You use the avSignalEvent method when there is an event you would like other components within the Framework to be notified about:

- You place the event id to be signaled in the WithID parameter and any alphanumeric or numeric values you want to pass in the SendAInfon or SendNInfon parameters, where n is 1,...,5. In this example the event is EMPLOYEE_CHANGED and the employee number is the value to be passed.
- By default the value of the To parameter is FRAMEWORK which means any active component in the framework will receive this signal and will need to test to see if it pertains to them. If you know that the event only pertains to the business object in which this component resides, you should set the parameter To equal to BUSINESSOBJECT so that a very limited set of components are notified of this event. Using this technique will improve performance of your

application. For a mini filter you need to signal at FRAMEWORK level.

4. Compile the command handler.

Step 3. Add a Routine to Listen for the EMPLOYEE_CHANGED Event

In this step you will add an event routine to the Employees filter to listen for the EMPLOYEE_CHANGED event.

1. Locate and open the Employees filter iiiCOM07.
2. Display the Source tab.
3. Add an event routine to listen for an avEvent signal:

```
EvtRoutine Handling(#Com_Owner.avEvent) WithId(#EventId)
Options(*NOCLEARMESSAGES *NOCLEARERRORS)
WithAinfo1(#Ainfo1)

EndRoutine
```

The avEvent event is an event that is signaled by the framework when the avSignalEvent method is called. The event routine receives the event being signaled along with any alphanumeric or numeric data that accompanies the event.

4. Next add a CASE command to test which event has been signaled in the event routine:

```
Case #EventId.Value

EndCase
```

It is important to test to make sure that the EventID matches one of the events that this event routine handles as the event being signaled might not apply to the component in which you have placed this event routine.

5. Then place appropriate WHEN VALUE_IS commands to handle various event ids that apply to this component. In this case we only want to handle the EMPLOYEE_CHANGED event:

```
When (= EMPLOYEE_CHANGED)
```

Your code will now look like this:

```

Evtroutine Handling(#Com_Owner.avEvent) WithId(#EventId) Options(*NOCLEARMESSAGES *NOCLEARERRORS)
  WithAinfo1(#ainfo1)
  Case #EventId.Value
    When (= EMPLOYEE_CHANGED)
  EndCase
Endroutine
End_Com
```

6. Next add code to save the current key values from overwrites:

```
Inz_List #Save_Keys 1
```

7. Assign the value passed by the signaled event to the EMPNO field:

```
#EmpNo := #AInfo1
```

8. Start updating the instance list:

```
Invoke Method(#avListManager.BeginListUpdate)
```

9. Fetch the details of the employee that has been updated:

```
FETCH FIELDS(#SURNAME #GIVENAME #EMPNO #SALARY)
FROM_FILE(PSLMST) WITH_KEY(#EMPNO)
```

10. Update the entry in the instance list:

```
Use Builtin(BCONCAT) With_Args(#GIVENAME #SURNAME)
To_Get(#FULLNAME)
```

```
Invoke #avListManager.AddtoList Visualid1(#EMPNO)
Visualid2(#FULLNAME) NColumn1(#SALARY) AKey1(#EMPNO)
```

11. Complete the instance list update:

```
Invoke Method(#avListManager.EndListUpdate)
```

12. Lastly restore the saved key values:

```
Get_Entry 1 #Save_Keys
```

Your finished code will look like this:

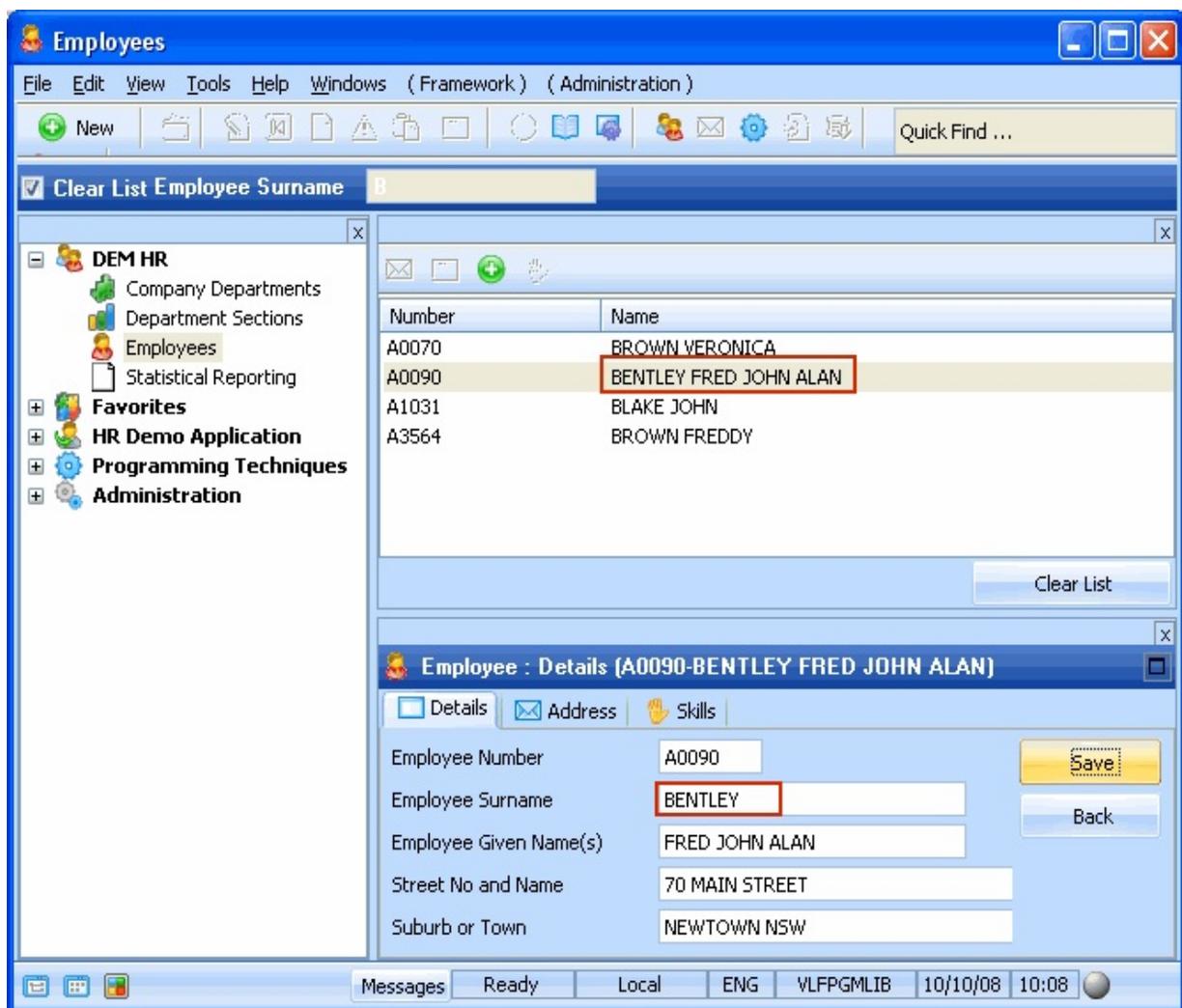
```
Evtroutine Handling(#Com_Owner.avEvent) WithId(#EventId) Options(*NOCLEARMESSAGES *NOCLEARERRORS)
  WithAinfo1(#ainfo1)
  Case #EventId.Value
  -When (= EMPLOYEE_CHANGED)
    Inz_List #Save_Keys 1
    #EmpNo := #AInfo1
    Invoke Method(#avListManager.BeginListUpdate)
    FETCH FIELDS(#SURNAME #GIVENAME #EMPNO) FROM_FILE(PSLMST) WITH_KEY(#EMPNO)
    Use Builtin(BCONCAT) With Args(#GIVENAME #SURNAME) To Get(#FULLNAME)
    Invoke Method(#avListManager.EndListUpdate)
    Get_Entry 1 #Save_Keys
  -EndCase
-Endroutine
```

13. Compile the filter.

Step 4. Test Signaling

In this step you will test the signaling of the EMPLOYEE_CHANGED event.

1. Start the Framework.
2. Select the iii DEM application and the Employees business object.
3. Use the filter to populate the instance list.
4. Select an employee and change the surname.
5. Click Save. Notice that the filter now listens for the EMPLOYEE_CHANGED event and updates the list entry:



Summary

Important Observations

- The Framework manager provides a simple to use event signaling service that may be used in Windows or Web browser applications.
- To make event-processing work you need a filter or command handler that signals the event and other filters or command handlers that listen for the event. Additional information may be sent along with the event.
- To update an instance list entry you use the `#ListManager.UpdateListEntryData` method.

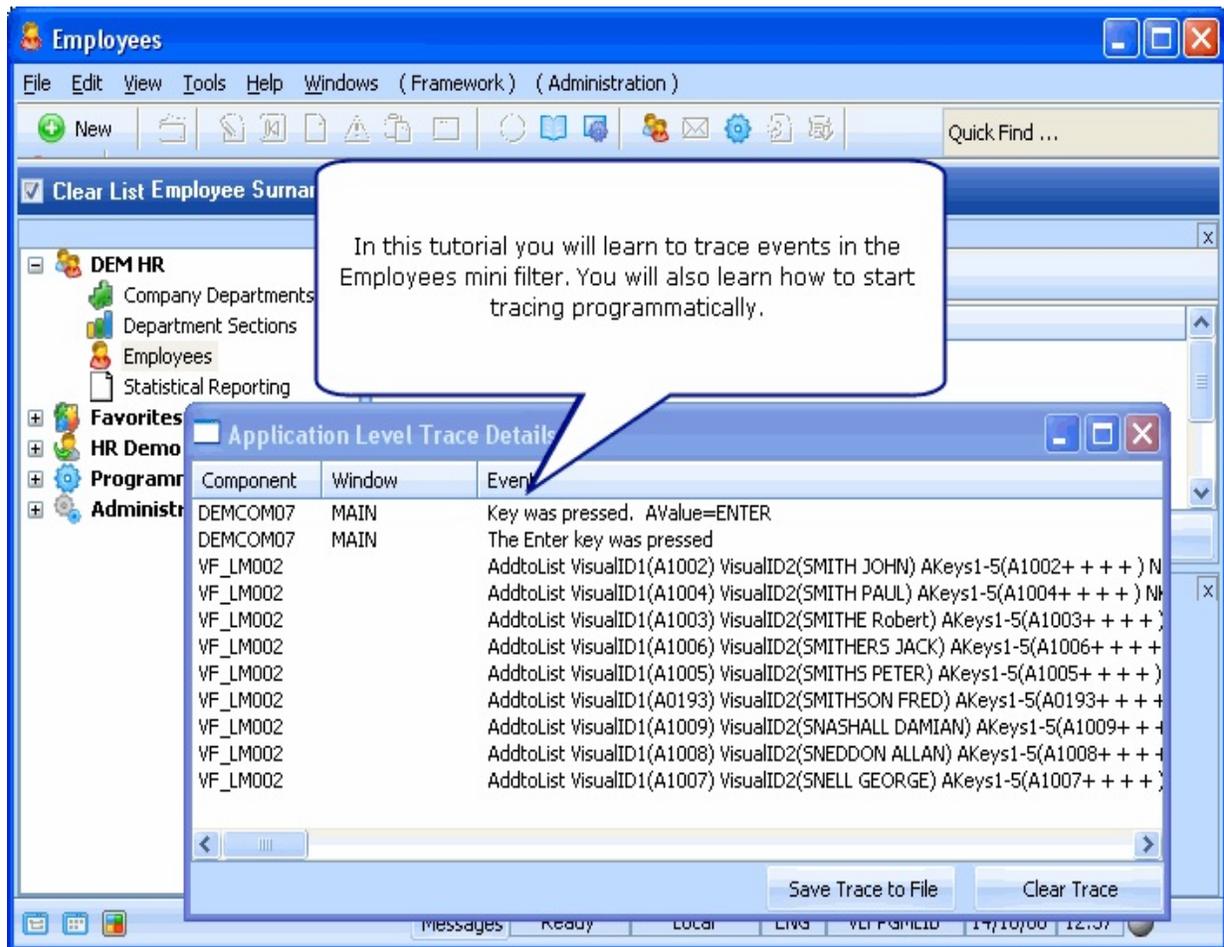
Tips & Techniques

What I Should Know

- How to signal an event
- How to listen for a signaled event
- How to update an entry in the instance list

VLF014WIN - Debugging/Tracing Objectives

- To learn how to use the tracing service to help you locate problems in your filters or command handlers. (See [Basic Tracing Service](#).)



To achieve this objective, you will complete the following steps:

[Step 1. Add a Trace Statement to Indicate Enter Key Was Pressed](#)

[Step 2. Add More Trace Statements](#)

[Step 3. Start Tracing Programmatically](#)

[Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

- Tutorials VLF000 – VLF007xxx.

Step 1. Add a Trace Statement to Indicate Enter Key Was Pressed

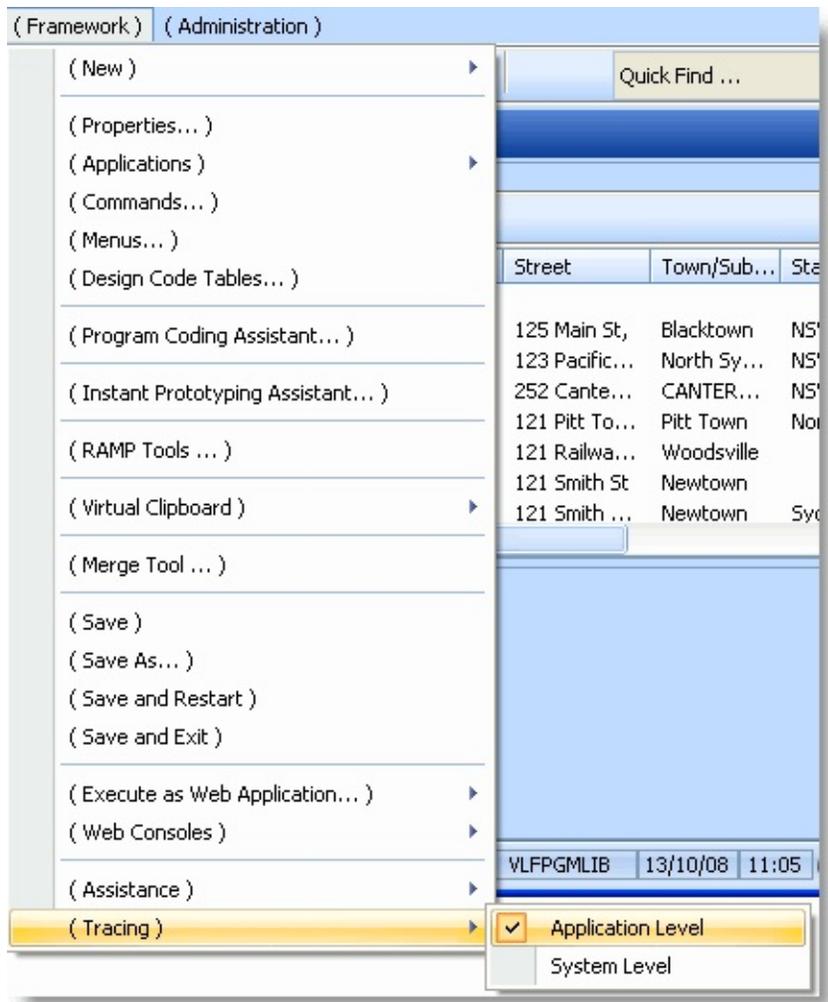
In this step, you will add a trace statement to show when the Enter key was pressed in the Employees mini filter.

1. In the Visual LANSA editor open the Employees mini filter iiiCOM07.
2. Display the Source tab.
3. Locate the event routine handling the #SURNAME.KeyPress event.
4. After the IF statement testing if the Enter key was pressed add this tracing command:

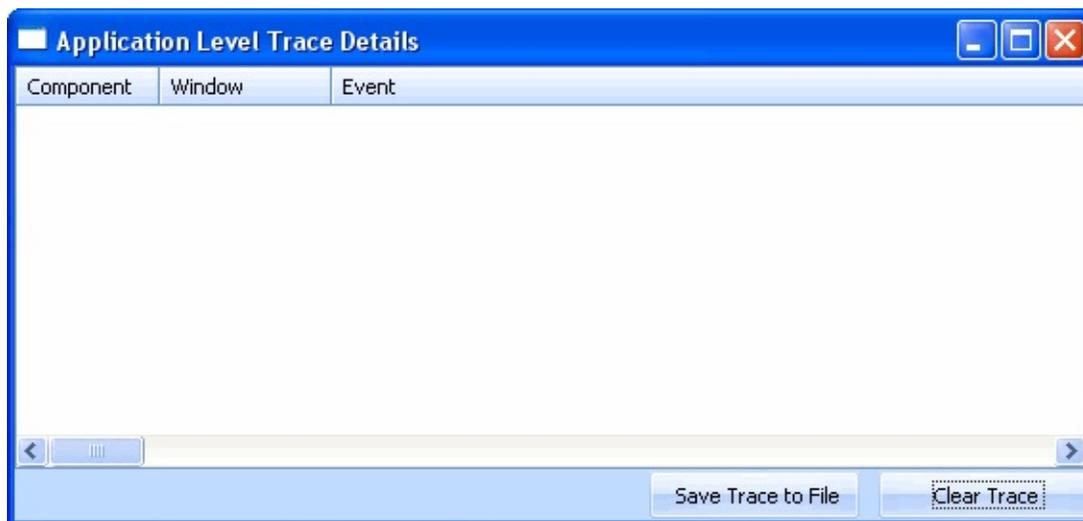
```
Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner)  
Event('The Enter key was pressed')
```

```
EVTRoutine HANDLING(#SURNAME.KeyPress) OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES) KeyCode(#keycode)  
  If ('#keycode.value = Enter')  
    Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner) Event('The Enter key was pressed')  
  Inz_List #Save_Keys 1
```

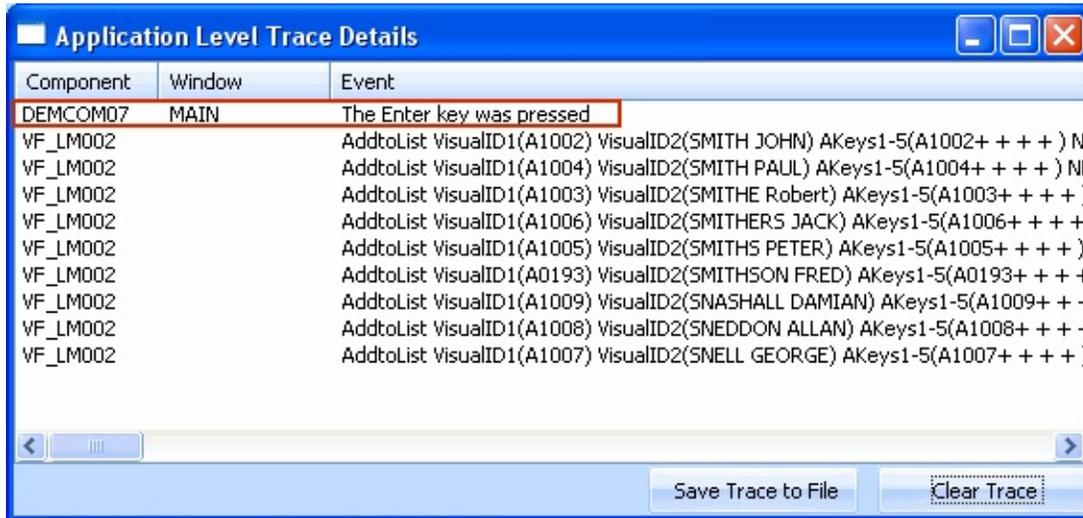
5. Compile the filter.
6. Start the Framework.
7. In the Framework menu select the Tracing option and then Application Level.



The Trace Details window is displayed:



8. Expand the iii HR application and select the Employees business object. (Move the tracing window to the side if necessary).
9. Move the focus to the mini filter and press Enter. The tracing window will now show that the Enter key was pressed.



10. Close the Framework.

Step 2. Add More Trace Statements

In this step, you will add two more trace statements to the filter.

The first tracing statement will show which key was pressed in the surname field. The second statement will show when the EMPLOYEE_CHANGED event was signalled and the employee number passed.

1. Just after the EVTROUTINE HANDLING(#SURNAME.KeyPress) event add this statement to trace the value of the key that was pressed:

```
#AVFRAMEWORKMANAGER.avRecordTraceAValue  
Component(#COM_OWNER) AValue(#keycode) Event('Key was pressed')
```

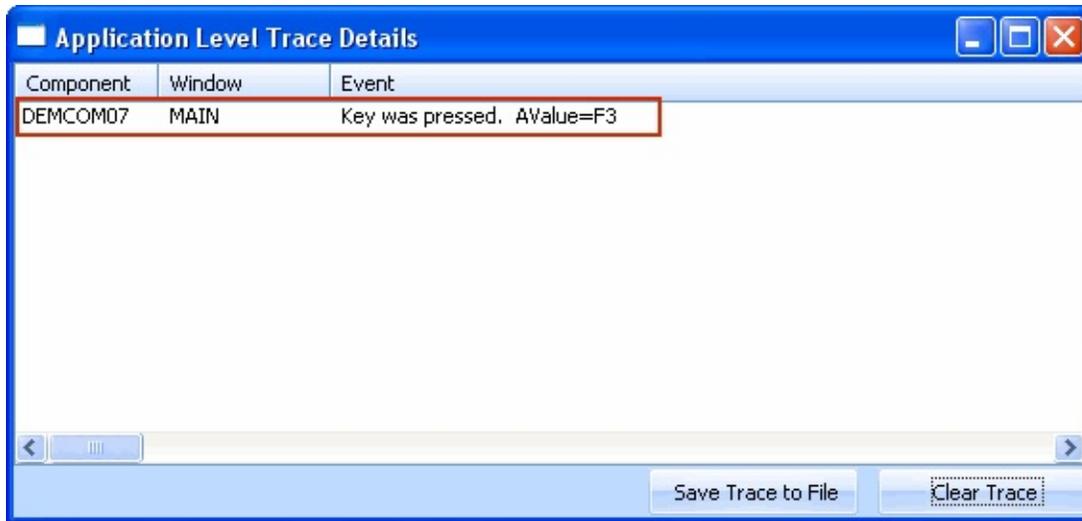
```
EVTROUTINE HANDLING(#SURNAME.KeyPress) OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES) KeyCode(#keycode)  
#AVFRAMEWORKMANAGER.avRecordTraceAValue Component(#COM_OWNER) AValue(#keycode) Event('Key was pressed')  
  
if ('#keycode.value = Enter')  
    Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner) Event('The Enter key was pressed')
```

2. Now locate the event routine handling the #Com_Owner.avEvent and add this statement to trace when the EMPLOYEE_CHANGED event is triggered and to show the employee number passed by the event:

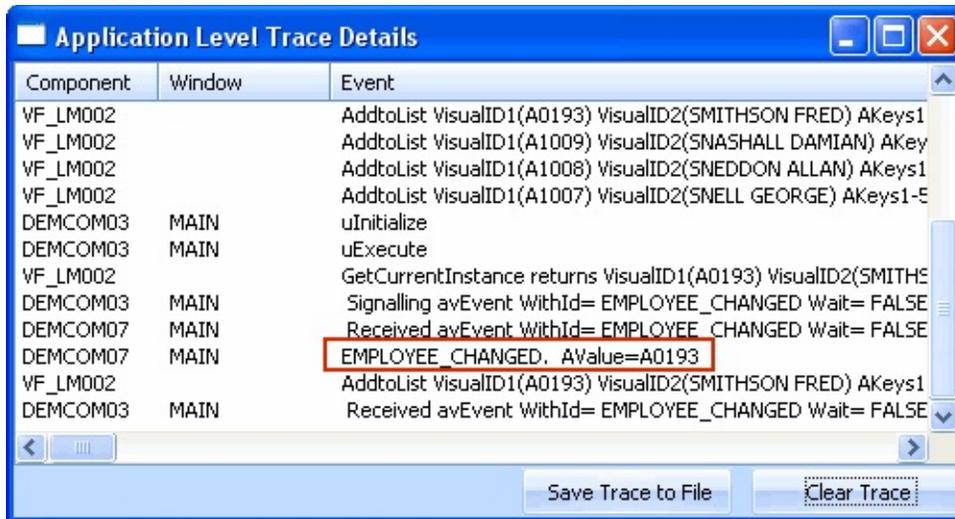
```
#AVFRAMEWORKMANAGER.avRecordTraceAValue  
Component(#COM_OWNER) AValue(#EMPNO)  
Event(EMPLOYEE_CHANGED)
```

```
EvtRoutine Handling(#Com_Owner.avEvent) WithId(#EventId) Options(*NOCLEARMESSAGES *NOCLEARERRORS)  
    WithAInfo1(#ainfo1)  
  
Case #EventId.Value  
    When (= EMPLOYEE_CHANGED)  
        Inz_List #Save_Keys 1  
  
        #EmpNo := #AInfo1  
  
        #AVFRAMEWORKMANAGER.avRecordTraceAValue Component(#COM_OWNER) AValue(#EMPNO) Event(EMPLOYEE_CHANGED)
```

3. Compile the filter.
4. Start the Framework and expand the iii HR application.
5. Turn the application level tracing on using the Framework menu.
6. Move the focus to the mini filter and press F3. The tracing window shows the key pressed.



7. Next select employees using the filter and then select one employee to display the Employee Details command handler.
8. Change one of the employee details and click the Save button.
9. Notice the EMPLOYEE_CHANGED event and the employee number are shown in the trace:



10. Close the Framework.

Step 3. Start Tracing Programmatically

In this step you will add code to start tracing when the Employee filter is used and end the trace when the filter is no longer used.

1. Add this statement to the uInitialize method routine to start the trace when the filter is initialized:

```
Set #AvFrameworkManager avTrace(TRUE)
```

```
Mthroutine Name(uInitialize) Options(*Redefine)
  Set #AvFrameworkManager avTrace(TRUE)
  #COM_OWNER.avMiniFilter := true
  #COM_OWNER.avMiniFilterpanel <= #PANL_1
  Invoke #AvFrameworkManager.avRestoreValue WithID1(SURNAME) ToAValue(#SurName)
Endroutine
```

2. Add a uTerminate method to the filter and a statement to terminate the trace when the filter is no longer used:

```
MTHROUTINE NAME(uTerminate) OPTIONS(*REDEFINE)

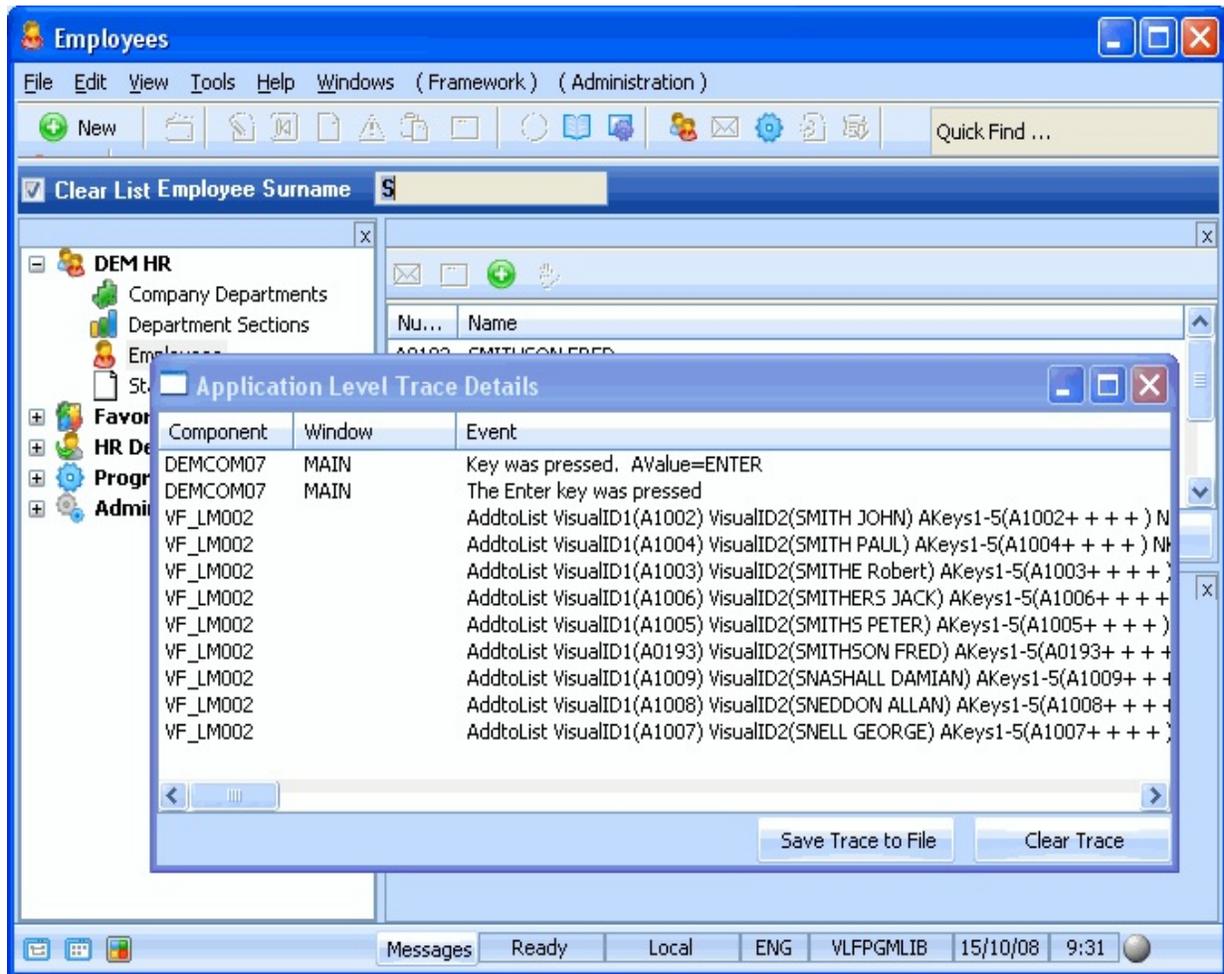
Set #AvFrameworkManager avTrace(False)

ENDROUTINE
```

```
MTHROUTINE NAME(uTerminate) OPTIONS(*REDEFINE)
  Set #AvFrameworkManager avTrace(False)
ENDROUTINE
```

3. Compile the filter.
4. Start the Framework.
5. Select the iii HR application and then the Employees business object. Notice that the tracing window is now displayed.

6. Select employees using the filter:



7. Select another business object or application. Notice that the tracing window is closed because the uTerminate method stopped tracing when the filter was no longer used.

8. Close the Framework.

Summary

Important Observations

- The Framework manager provides a basic tracing service to help you locate problems in your filters or command handlers.
- The tracing service can be used in conjunction with, or independently of, the normal LANSa application debugging and tracing facilities.

•

Tips & Techniques

- You can leave these method calls inside your code. The only time they have any effect is if tracing is turned on. Implementing tracing using this method is ideal as you don't have to remove the code at all if you do not wish to do so.
- The trace information can give you a lot of detailed information about what has happened which saves you having to run your application in debug mode.
- The first column in the tracing window contains the component name, so you always know which component the traced event is associated with.

What I Should Know

- How to trace Framework applications
- How to trace specific events in a filter or command handler
- How to start and stop tracing programmatically

Tutorials for WAM Web Browser Applications

Applies to **WAM** only.

Includes:

[VLF006WAM - Snapping in A Real WAM Web Filter](#)

[VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

[VLF009WAM - Adding Instance List Columns in WAM Applications](#)

[VLF011WAM - Creating a Parent Child Instance List](#)

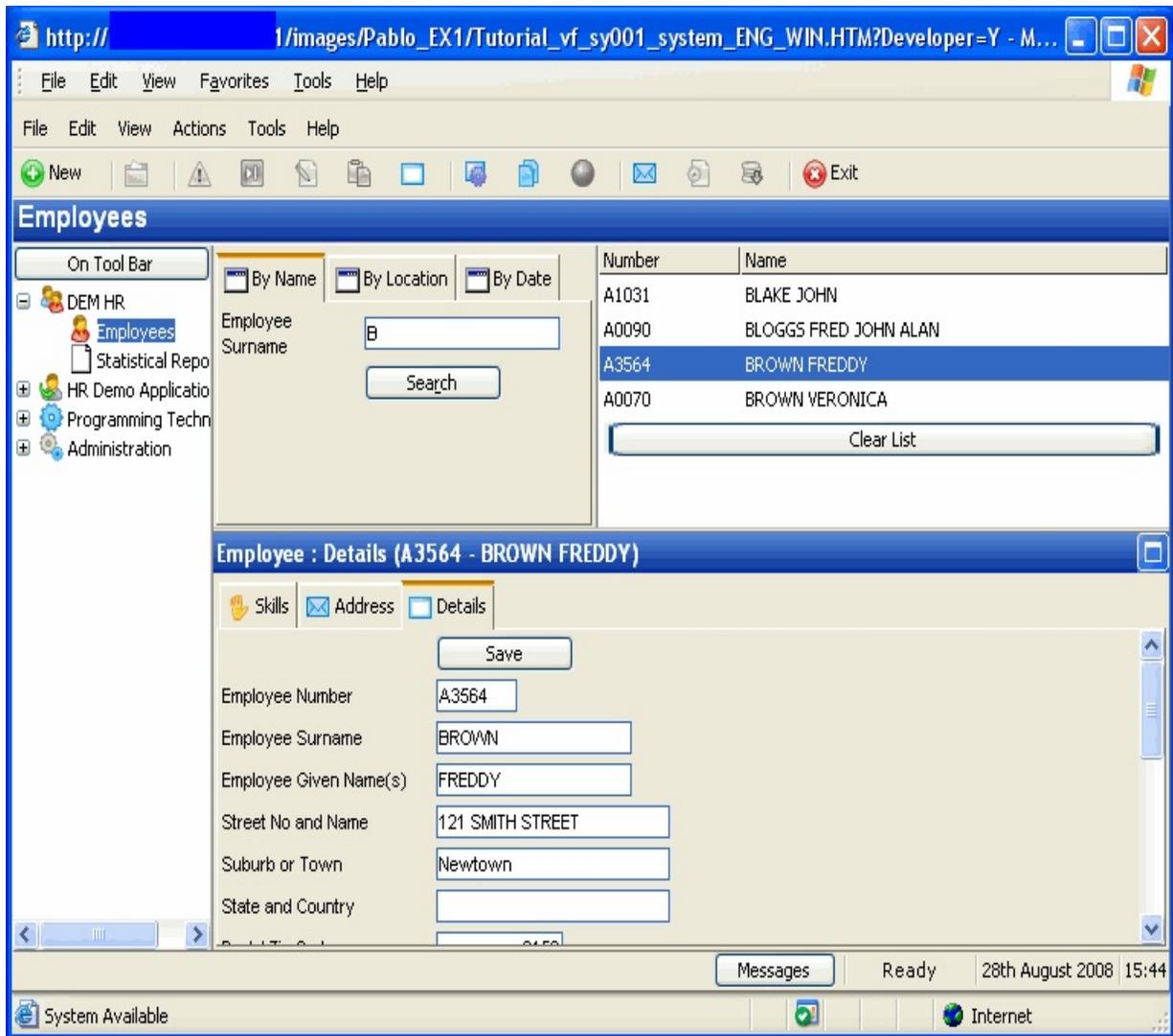
[VLF012WAM - Controlling Navigation Using Switching and the Virtual Clipboard](#)

[VLF013WAM - Signaling Events](#)

[VLF014WAM - Debugging/Tracing](#)

After you have created and validated your prototype, you can develop it into a finished application. The basic structure and presentation of the application will remain unchanged as you continue to use the Framework. To complete the application, you simply replace the prototype filters and command handlers with real WAM ones.

In these tutorials, you will replace the employee filters with real filters and the Details prototype command handler with a real command handler:



These tutorials assume you are working on a LANSAslave system connected to a server. If you are using an independent LANSAsystem, please ignore the steps to check objects into the server.

The Personnel File

When prototyping your application, you decide your business objects based on an analysis of the tasks of the users of your application. At that point the database structure is not important.

Now that you are about to start implementing real filters and command handlers, you need to know how the data you will be using is stored.

The following tutorials are based on the PSLMST Personnel demonstration file. Locate this file in the repository and view its properties:

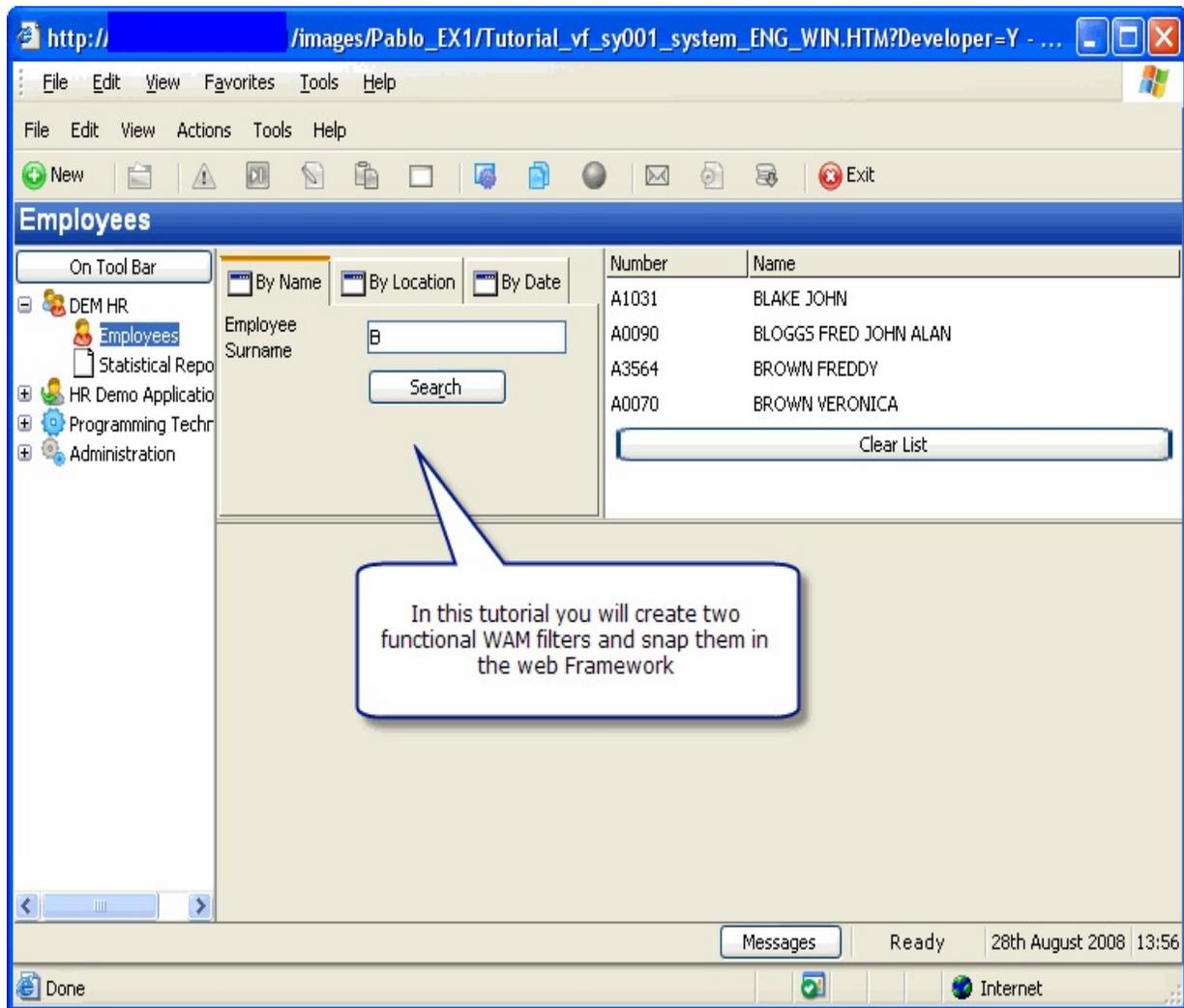
The screenshot shows a window titled "Properties of file PSLMST" with a tree view on the left and a list of field definitions on the right. The tree view is expanded to show "Real fields (14)", "Virtual fields", and "Logical views".

Field Name	Description	Format
Real fields (14)		
EMPNO	Employee Number	Alpha 5
SURNAME	Employee Surname	Alpha 20
GIVENAME	Employee Given Name(s)	Alpha 20
ADDRESS1	Street No and Name	Alpha 25
ADDRESS2	Suburb or Town	Alpha 25
ADDRESS3	State and Country	Alpha 25
POSTCODE	Post / Zip Code	Signed 6,0
PHONEHME	Home Phone Number	Alpha 15
PHONEBUS	Business Phone Number	Alpha 15
STARTDTER	Start date (YYMMDD)	Signed 6,0
TERMDATER	Termination Date (YYMMDD)	Signed 6,0
DEPARTMENT	Department Code	Alpha 4
SECTION	Section Code	Alpha 2
SALARY	Employee Salary	Packed 11,2
Virtual fields		
STARTDTE	Start Date (DDMMYY)	Signed 6,0
TERMDATE	Termination Date (DDMMYY)	Signed 6,0
MNTHSAL	Monthly Salary	Packed 11,2
Logical views		
PSLMST1	Personnel by Deptment, Section, Employee	
Non-unique key		
Keys		
DEPARTMENT	Ascending (Unsigned)	
SECTION	Ascending (Unsigned)	
EMPNO	Ascending (Unsigned)	
PSLMST2	Personnel by Surname, Given Name	
Non-unique key		
Keys		
SURNAME	Ascending (Unsigned)	
GIVENAME	Ascending (Unsigned)	

Close

VLF006WAM - Snapping in A Real WAM Web Filter Objective

- Learn how to replace prototype filters with real WAM filters. These will perform the actual selection of the items for the [Instance List](#) when the Framework is running in web mode.



To achieve this objective, you will complete the following steps:

- Step 1. [Creating Your Real WAM Filter](#)
- Step 2. [Snapping In the WAM By Name Filter](#)
- Step 3. [Creating a WAM By Location Filter](#)
- Step 4. [Snapping in the WAM By Location Filter](#)
- Summary

Before You Begin

You may wish to review:

- [Filters](#) in [Key Concepts](#)
- [Framework Programming](#).

In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)

Step 1. Creating Your Real WAM Filter

In this step, you will create your own filter by creating a WAM which will be snapped into the Visual LANSA Framework.

1. Make sure that the Enable Framework for WAMs option in the Framework Details tab is selected. In the Visual LANSA editor check that you can create new WAMs (the option is available if your system is enabled for Web).
2. Start the Program Coding Assistant in the Framework using the (Framework) menu.

The Program Coding Assistant window is displayed. It allows you to create different types of components that can be plugged into your filters, instance lists and command handlers. It is highly recommended to use the program coding assistant when you first start using the Framework.

Initially you will most likely use filters that generate a component that can be executed (e.g. CRUD Filter (Create/Read/Update/Delete), Filter that searches a file or view). As you progress you might only use a skeleton filter or simply copy from one that is similar to one that you want to create.

3. If you are using a non-English system, click on Framework -> Your Framework in the top-left tree view. The Set LANSA code generation preferences option appears at the bottom. Select this option and set your preferences.
4. Select the iii HR application and then the By Name filter.
5. Choose Web – using WAM components as the platform and and Filter that searches using a file or a view.

Program Coding Assistant

Select the object you want to generate code for

- Application->HR Demo Application
- Application->Administration
- Application->DEM HR
 - Business Object->Employees
 - Command Handler->Details
 - Command Handler->Skills
 - Command Handler->Address
 - Command Handler->New
 - Filter->By Name
 - Filter->By Location
 - Filter->By Date
- Business Object->Statistical Reporting
- Command Handler->About

Refresh

Select the platform you want to generate for

- Native MS Windows
- Web - using *WEBEVENT functions
- Web - using WAM components
- Web - using AJAX style components

Select the type of code you want to generate

CRUD Filter

Filter that searches using a file or view

A skeleton filter

Search button event handling routine (code fragment)

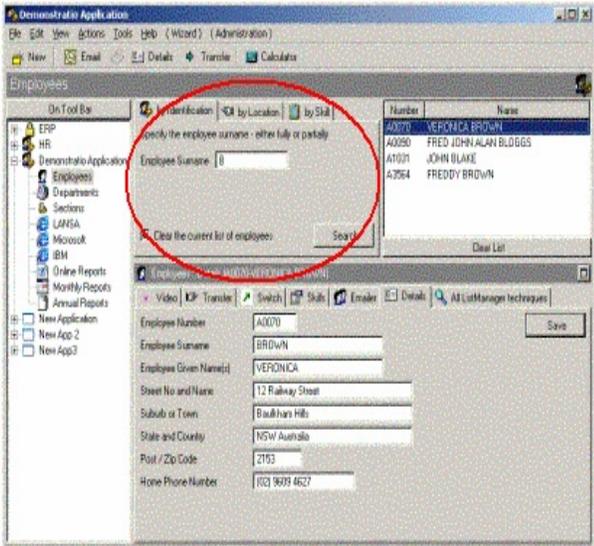
Invoke #avListManager.AddtoList (code fragment)

Filter that searches using a file or view

What? This assistant produces the code for a filter that searches for information using a specified physical file or logical view.

Filters are used to dynamically create business object *instance lists* (e.g. lists of Customers, lists of Products, lists of Orders, lists of Employees, etc).

Typically Visual LANSa Framework filters are presented like this example (in the area circled in red):

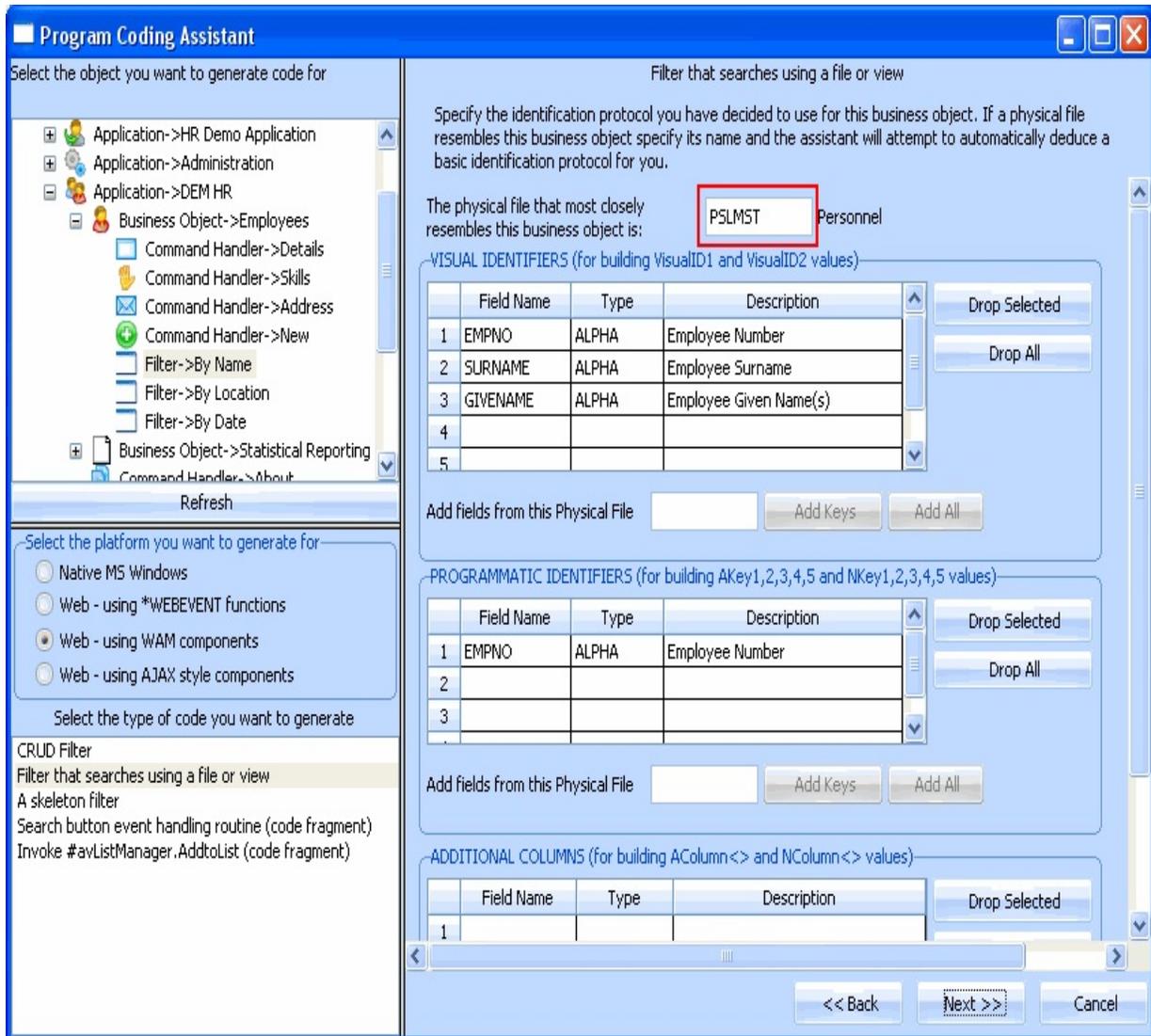


Number	Name
40070	VERONICA BROWN
40080	FRED JOHN ALAN BLOGGS
A1001	JOHN BLAKE
A3564	FREDDY BROWN

Next >> Cancel

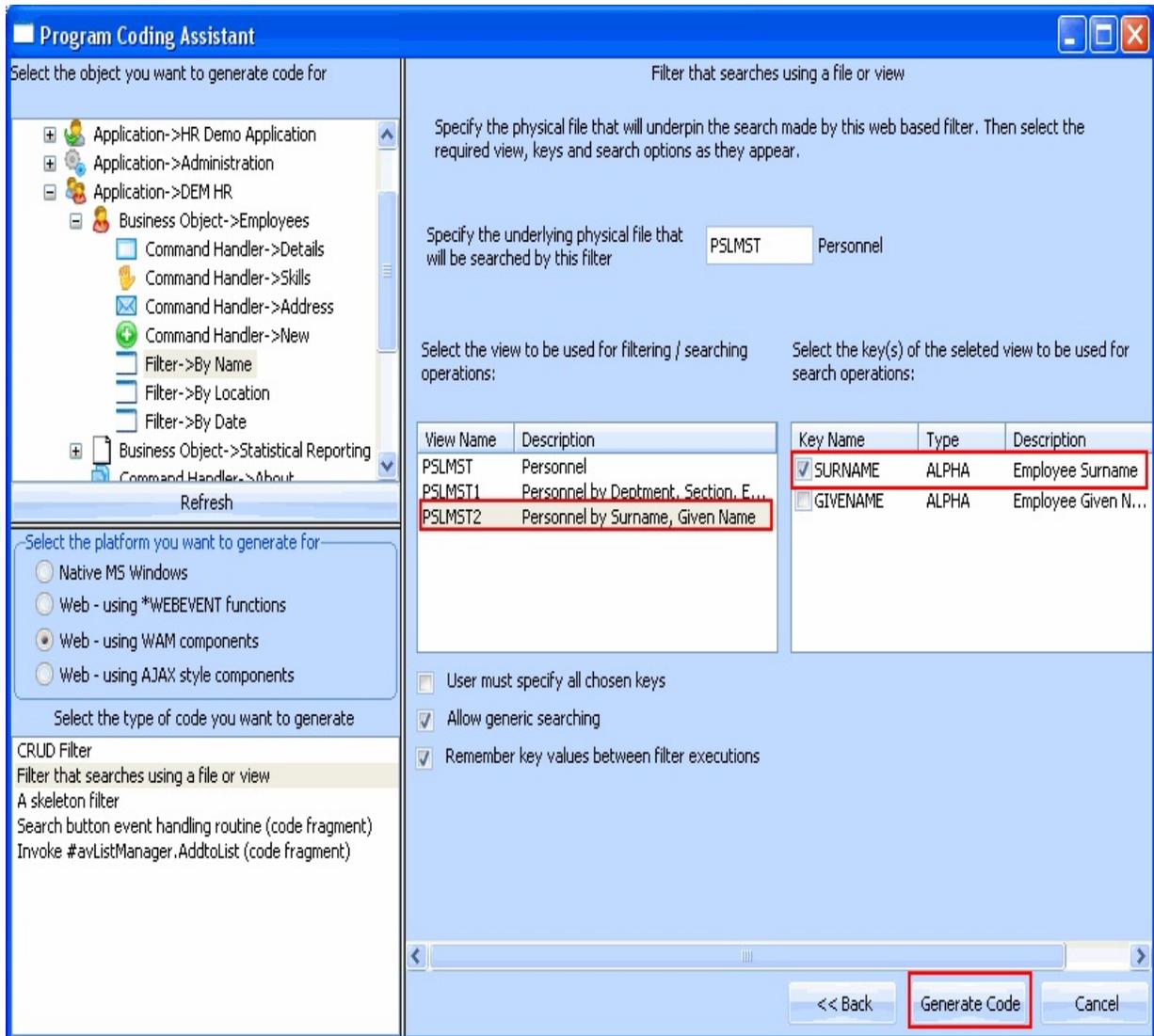
6. Click the Next button.

7. Specify PSLMST as the Physical File that most closely resembles this business object



The Program Coding Assistant detects the Visual and Programmatic Identifiers required.

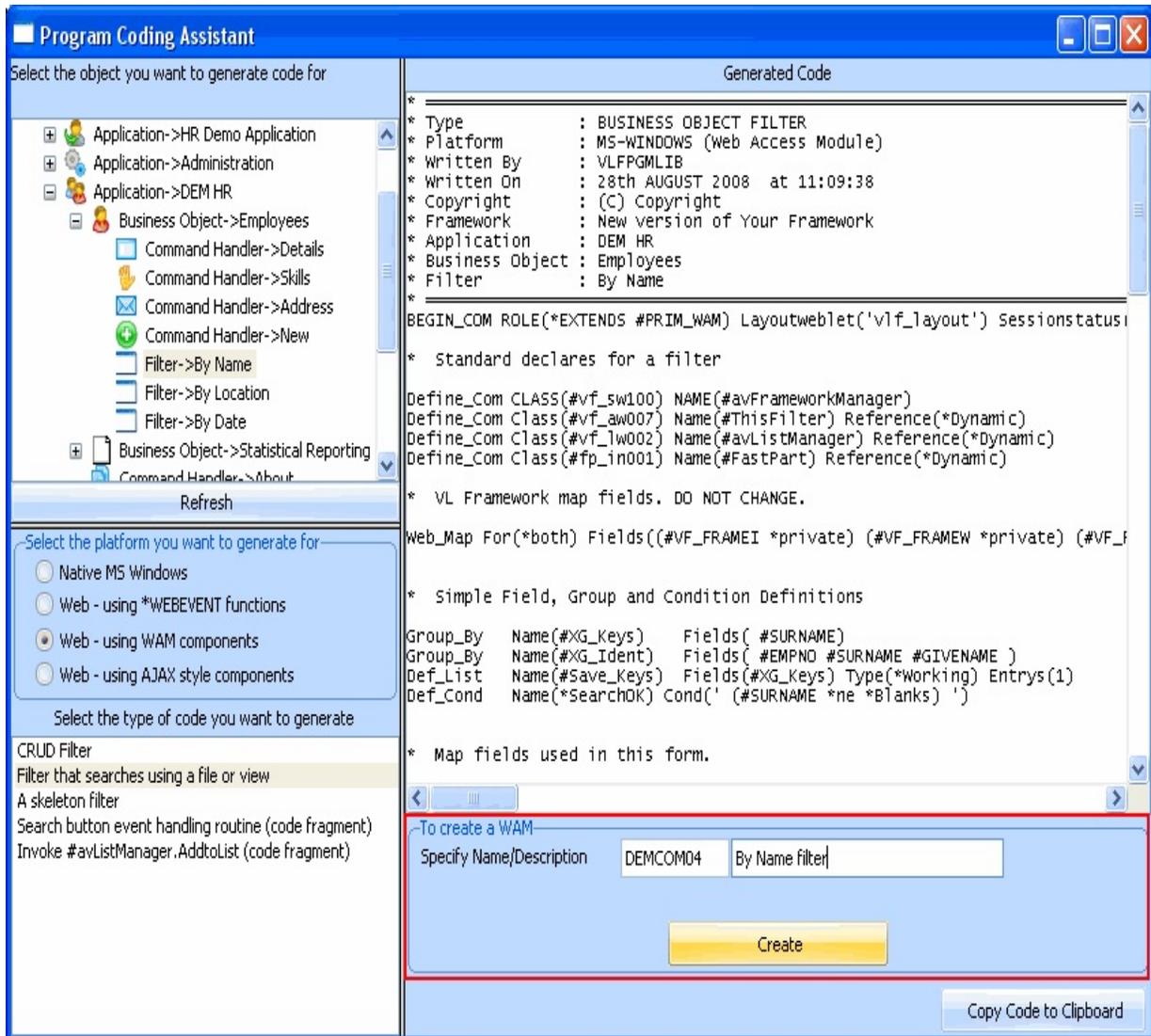
8. Click the Next button.
9. On the next page specify PSLMST2 as The view to be used for searching/filtering operations.
10. Specify SURNAME as the Key of the selected view to be used for search operations.



11. Click the Generate Code button.

The next page, Generated Code, displays the source code for your filter. You now need to create the component that will contain this code:

12. Specify iiiCOM04 as the name of your real filter and By Name Filter as the description. (iii are your initials If you are using an unlicensed or trial version of Visual LANSA, you must always use the 3 characters DEM to replace iii).
13. Click on the Create button to create the WAM.



After a brief delay a message is shown indicating the WAM has been created.

14. Switch to the Visual LANSA editor. Your filter is displayed in the Visual LANSA editor.

15. Examine the code:

This statement tells the Framework that new entries are about to be added to the instance list:

```
#avListManager.BeginListUpdate
```

This statement clears the instance list:

```
#avListManager.ClearList
```

This statement reads the records that match the surname entered by the user:

```
Select Fields(#XG_Ident) From_File(PSLMST2) With_key(#XG_Keys)
Generic(*yes) Nbr_Keys(*Compute)
```

This statement sets up the visual Identifier(s)

```
#UF_VisID1 := #EMPNO
#UF_VisID2 := #SURNAME
#UF_VISID2 := #UF_VISID2.BlankConcat(#GIVENAME )
```

This statement adds the data to the instance list

```
Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#EMPNO)
```

VisualId1 will be shown in column one of the instance list and VisualId2 will be shown in column two of the instance list. Akey1 is the key that uniquely identifies an employee (in this case the field is alphanumeric, so its Akey1, not Nkey1).

This statement tells the Framework that you have finished adding entries to the instance list:

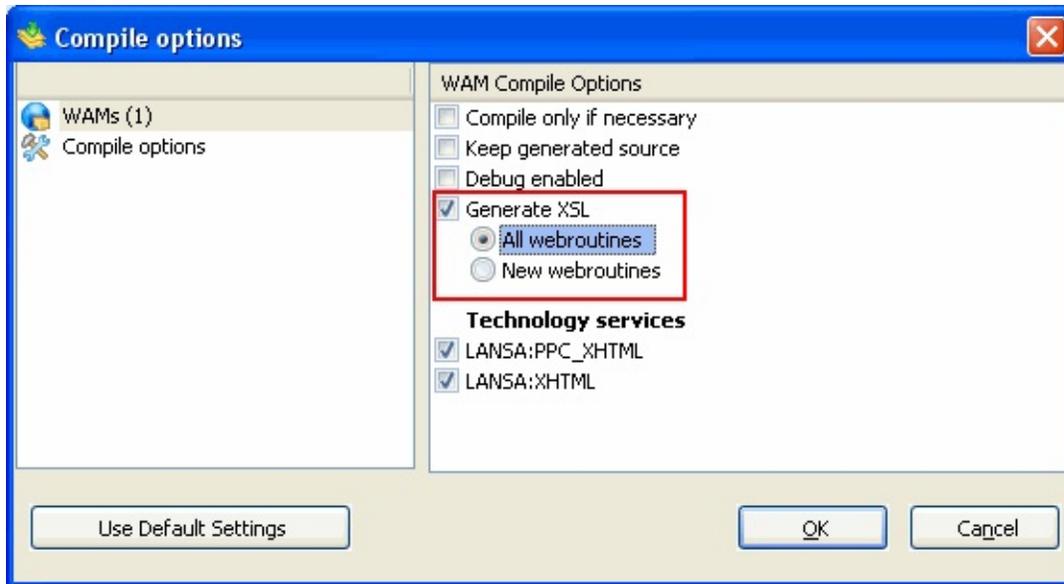
```
#avListManager.EndListUpdate
```

You may want to read [WAM Filter and Command Handler Anatomy](#) to see how WAMs are structured.

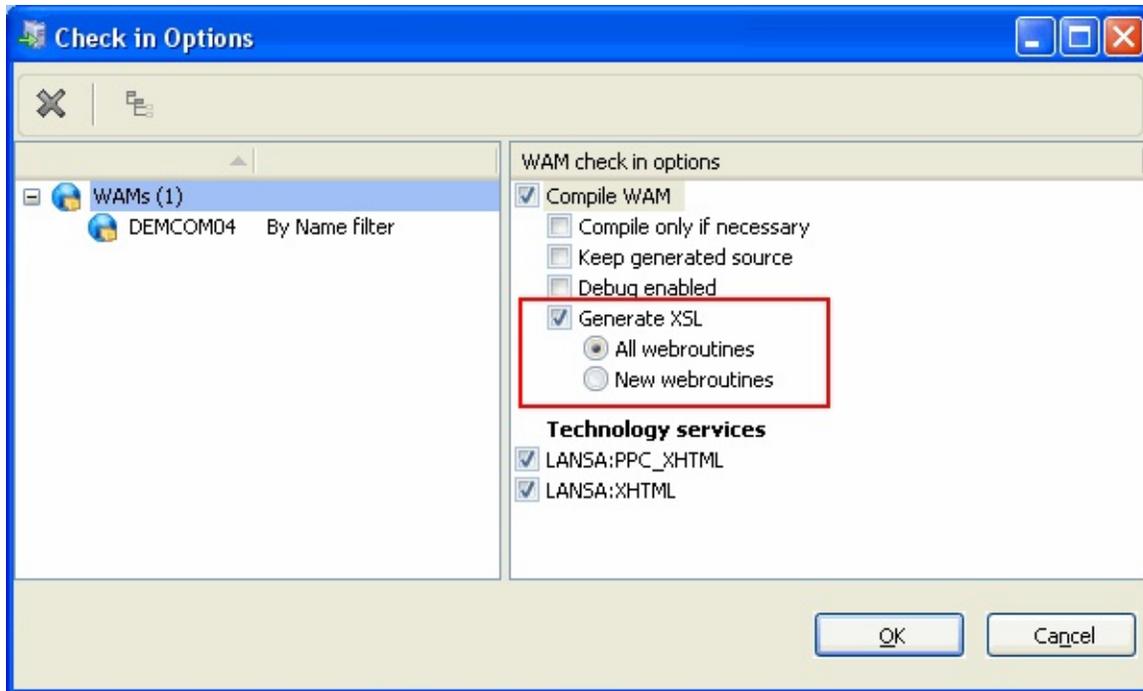
16. If you are using a partition with language NAT, you need to change the default value of the #UB_SEARCH field to SEARCH and change the Web_Map statement to:

```
Web_Map For(*both) Fields(#XG_Keys (#UB_SEARCH *Desc))
```

17. Select the Compile option in the Verify menu.
18. Ensure that the All webroutines option is selected for the Generate XSL option.



19. Ensure the compilation was successful.
20. Check in your changes to the server. First check in the layout weblet for your WAM::
 - a. Locate the layout weblet for your WAM under Weblets in the Repository. It is called iiiCOM04_layout (where iii correspond to your initials).
 - b. Right-click the weblet to bring up the associated pop-up menu and choose the Check in option.
 - c. Click OK.Then locate the WAM to check it in:
 - a. Right-click the WAM to bring up the associated pop-up menu and choose the Check in option.
 - b. Ensure that the All webroutines option is selected for the Generate XSL option.
 - c. Click OK to check the changes in.

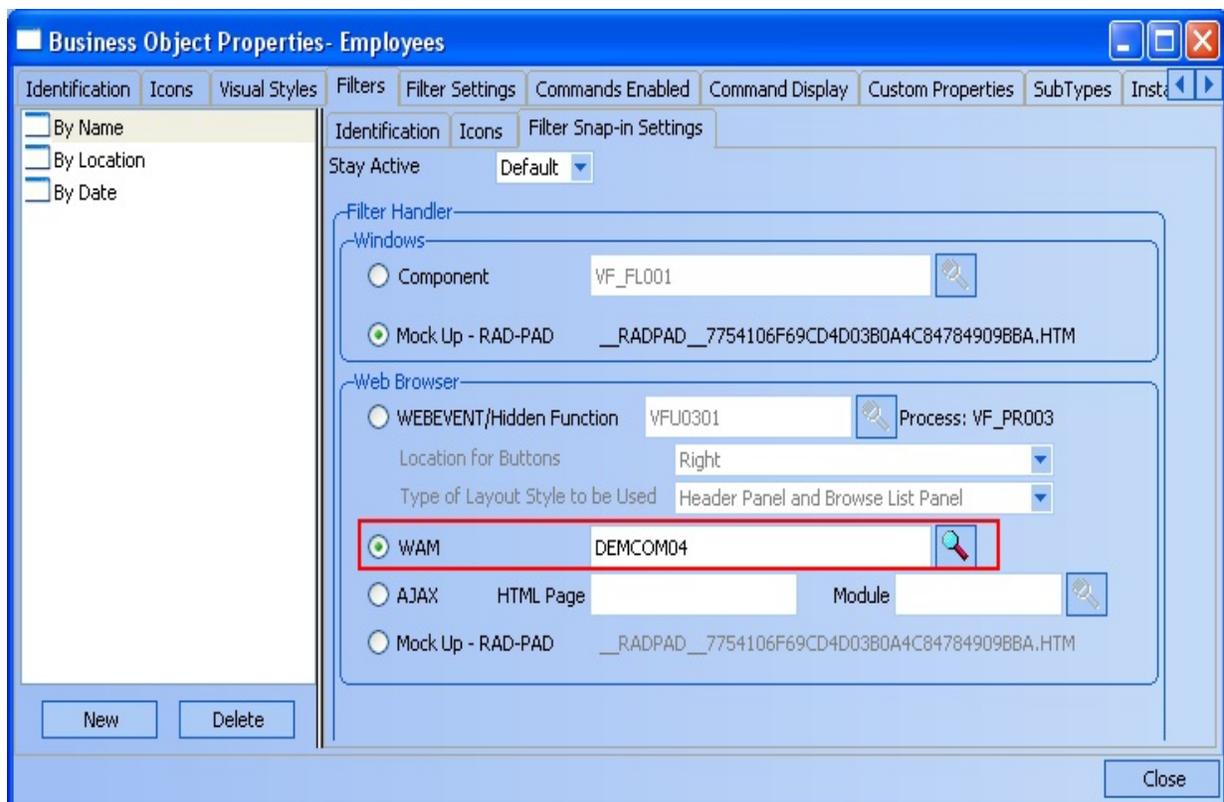


d. Wait until the compiles have finished.

Step 2. Snapping In the WAM By Name Filter

Now that you have compiled your new filter and are ready to test it, you need to snap it into the Framework.

1. In the Framework, close the Program Coding Assistant.
2. Select the iii HR application and double-click on the Employee business object.
3. On the resulting Business Object Properties dialog, click on the Filters tab.
4. Select the By Name filter. You will replace the web mock up filter with your real filter
5. Click on the Filter Snap-in Settings Tab.
6. Click on the WAM property radio button in the Web Browser group box.
7. Type the name of your WAM filter into the entry field.



8. Bring up the Instance List/Relations tab. Make the heading of the first column in the instance list Number (it will display employee numbers) and

the heading of the second column Name (it will display employee names).

Sequence	Type	Caption	Width % (Total 25%)	Decimals	Edit Code	Date/Time Output Format	ITC Converter
10	VISUALID1	Number	25		Default	SYSFMT8	Local -> Local
20	VISUALID2	Name			Default	SYSFMT8	Local -> Local
	ACOLUMN1				Default	SYSFMT8	Local -> Local
	ACOLUMN2				Default	SYSFMT8	Local -> Local
	ACOLUMN3				Default	SYSFMT8	Local -> Local
	ACOLUMN4				Default	SYSFMT8	Local -> Local
	ACOLUMN5				Default	SYSFMT8	Local -> Local
	ACOLUMN6				Default	SYSFMT8	Local -> Loc

Make sure the [Save and Restore Instance Lists](#) option is not selected.

Make sure the [Enable Clear List Button](#) is selected.

9. Use the (Framework) menu and select the option to save the Framework.

Accept the prompt to upload the Framework and wait while the upload completes.

10. Use the (Framework) menu and select the option to Execute as Web Application...

Accept the default options and press OK.

11. Select the iii HR application in the web Framework and then the Employees business object. Bring the By Name filter topmost. Type in a partial surname and click Search.

12. Your filter is now snapped into the Framework and usable.

http:// /images/Pablo_EX1/Tutorial_vf_sy001_system_ENG_WIN.HTM?Developer=Y - ...

File Edit View Favorites Tools Help

File Edit View Actions Tools Help

New [Icons] Exit

Employees

On Tool Bar

- DEM HR
 - Employees
 - Statistical Repo
- HR Demo Applicatio
- Programming Techn
- Administration

By Name By Location By Date

Employee Surname

Number	Name
A1031	BLAKE JOHN
A0090	BLOGGS FRED JOHN ALAN
A3564	BROWN FREDDY
A0070	BROWN VERONICA

Messages Ready 28th August 2008 13:44

Done [Icons] Internet

Step 3. Creating a WAM By Location Filter

In this step, you will create a real By Location filter that will locate Employees by the department and section in which they work.

1. Start the Program Coding Assistant.
2. Select the Framework object navigation tree in the upper left of the Program Coding Assistant form.
3. Drill down through the tree to find your By Location filter and select it.
4. Choose Web - using WAM components option as the platform.
5. Select the type Filter that searches using a file or view.
6. Click Next.

The Program Coding Assistant shows the PSLMST file as the physical file and detects the Visual and Programmatic Identifiers required. You do not need to change any of these values.

7. Click the Next button. On the screen:
 - Select the file view named PSLMST1 (Personnel by Department, Section, Employee Number).
 - Select the search keys DEPARTMENT and SECTION.
 - Uncheck User must specify all Chosen Keys.
 - Uncheck Allow Generic Searching.
 - Check Remember key values between filter executions.
8. Click the Generate Code button. The right hand side of the Program Coding Assistant now shows the code that it has generated for your filter.
9. In the Generated Code window specify iiiCOM05 (where iii are your initials) as the name of your filter and give it a description. If you are using an unlicensed or trial version of Visual LANSA, you must always use the 3 characters DEM to replace iii. Then click the Create button to create your filter.

(Alternatively you can copy the generated code to the clipboard by clicking the "Copy Code to Clipboard" button and paste into an existing WAM if you have created one.)
10. Your filter is displayed in the Visual LANSA editor. Compile it and choose to generate XSL for all web routines.

11. Check in the WAM and its associated layout weblet to the server.

Step 4. Snapping in the WAM By Location Filter

In this step, you will snap in your Location filter.

1. In the Framework, close the Program Coding Assistant.
2. Select the iii HR application and double-click on the Employees business object.
3. On the resulting Business Object Properties dialog, click on the Filters tab.
4. Select the By Location filter. You will replace the web mock up filter with your real filter.
5. Click on the Filter Snap-in Settings Tab.
6. Click on the WAM property radio button in the Web Browser group box.
7. Type the name of your WAM filter into the entry field.
8. Use the (Framework) menu and select the option to save the Framework.
Accept the prompt to upload the Framework and wait while the upload completes.
Use the (Framework) menu and select the option to Execute as Web Application...
Take the default options and press OK.
9. Your filter is now snapped into the Framework and usable.

Summary

Important Observations

- With snap-in real filters, you have now created real functionality in your application.

Tips & Techniques

- The source code for the filters used in the demonstration application can be found in the repository in components named DM_FILT*.

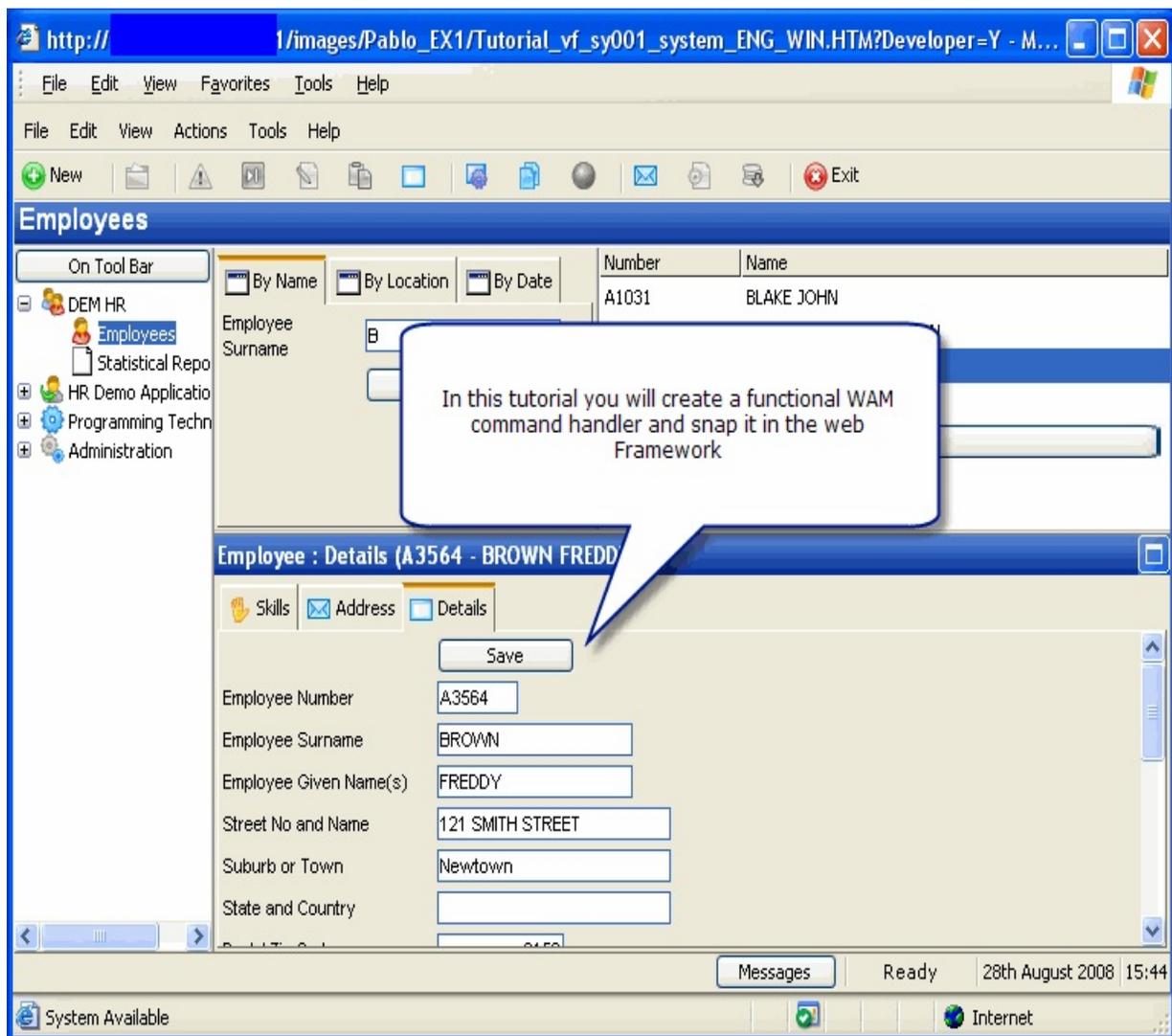
What I Should Know

- What you need to do to create your own WAM filters.
- How you snap them in the Framework.
- How to use the Program Coding Assistant.
- How to customize the way that instance lists are displayed.

VLF007WAM - Snapping in a Real WAM Web Command Handler

Objective

- Learn how to replace prototype command handlers with real web handlers which will perform actual processing when the Framework runs in web mode.
- To replace the Details prototype command handler with a real WAM command handler.



To achieve this objective, you will complete the following steps:

- [Step 1. Creating Your Real WAM Command Handler](#)
- [Step 2. Snapping in Your WAM Command Handler](#)
- [Summary](#)

Before You Begin

You may wish to review:

- [Command Handler](#)
- [Framework Programming.](#)

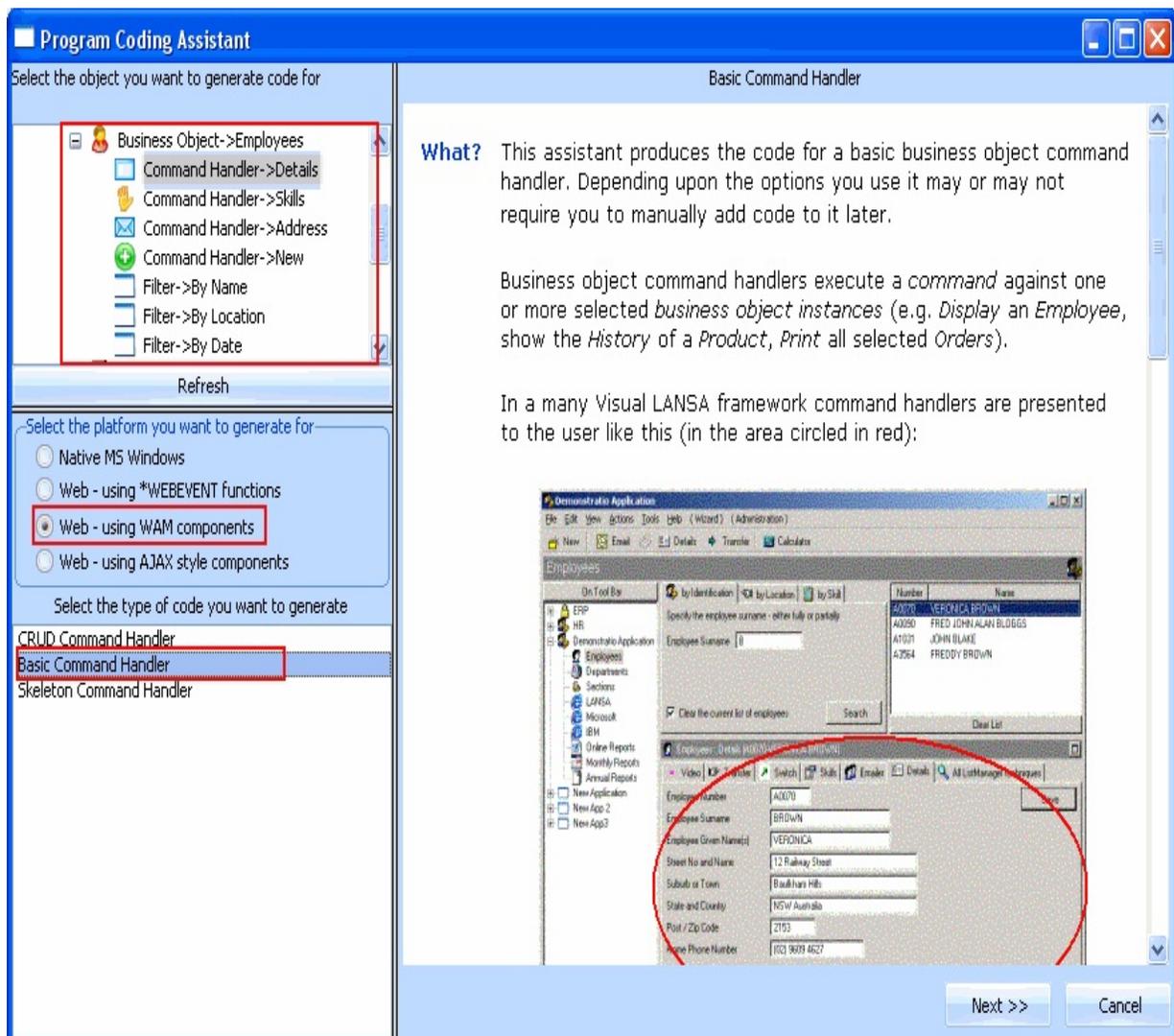
In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)
- [VLF006WAM - Snapping in A Real WAM Web Filter](#)

Step 1. Creating Your Real WAM Command Handler

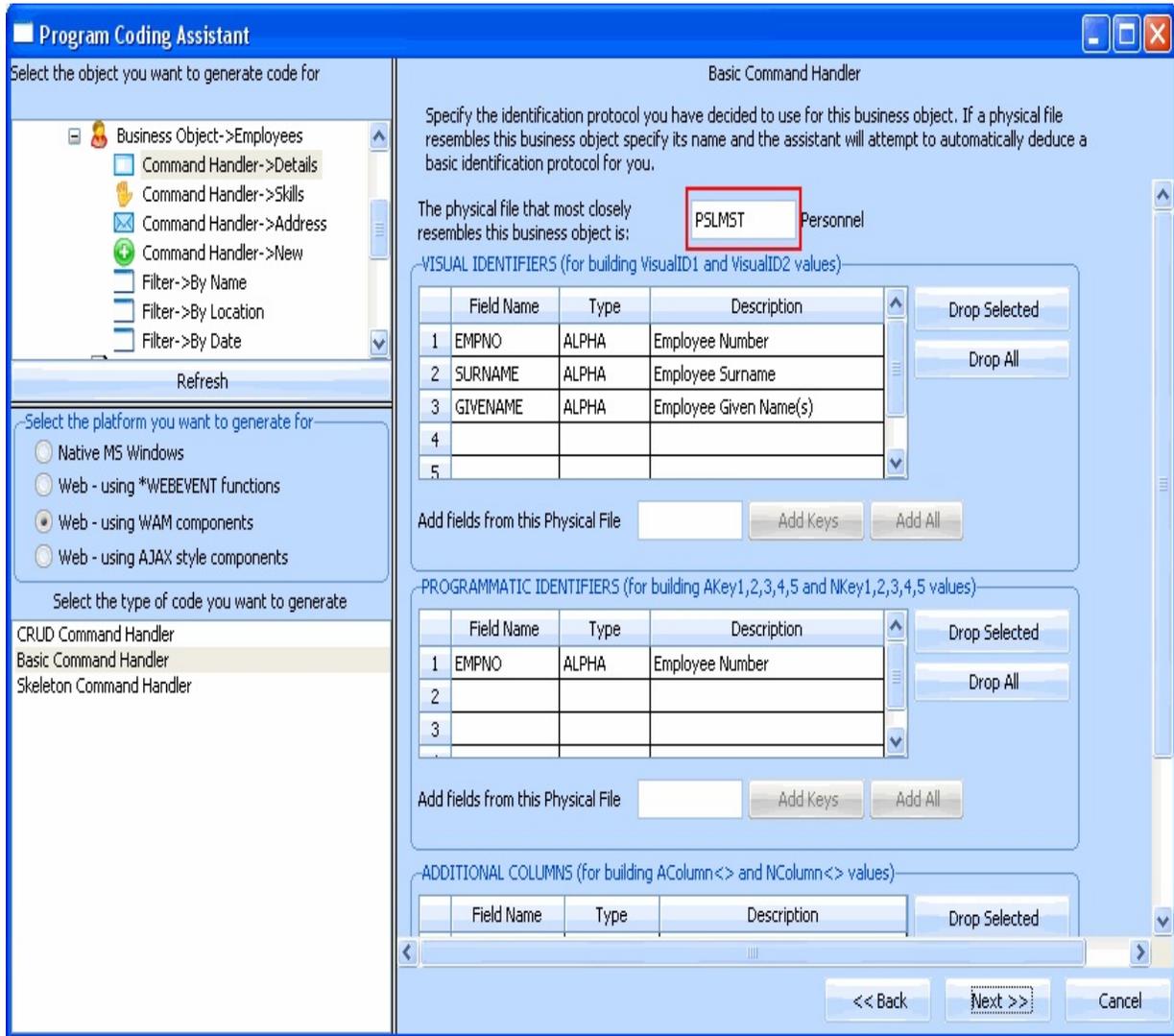
In this step, you will create a real WAM command handler for the Details command.

1. Start the Program Coding Assistant.
2. Select the iii HR application, then the Details command handler.
3. Select Web – using WAM components as the platform.
4. Select Basic Command Handler as the type of code you want create.



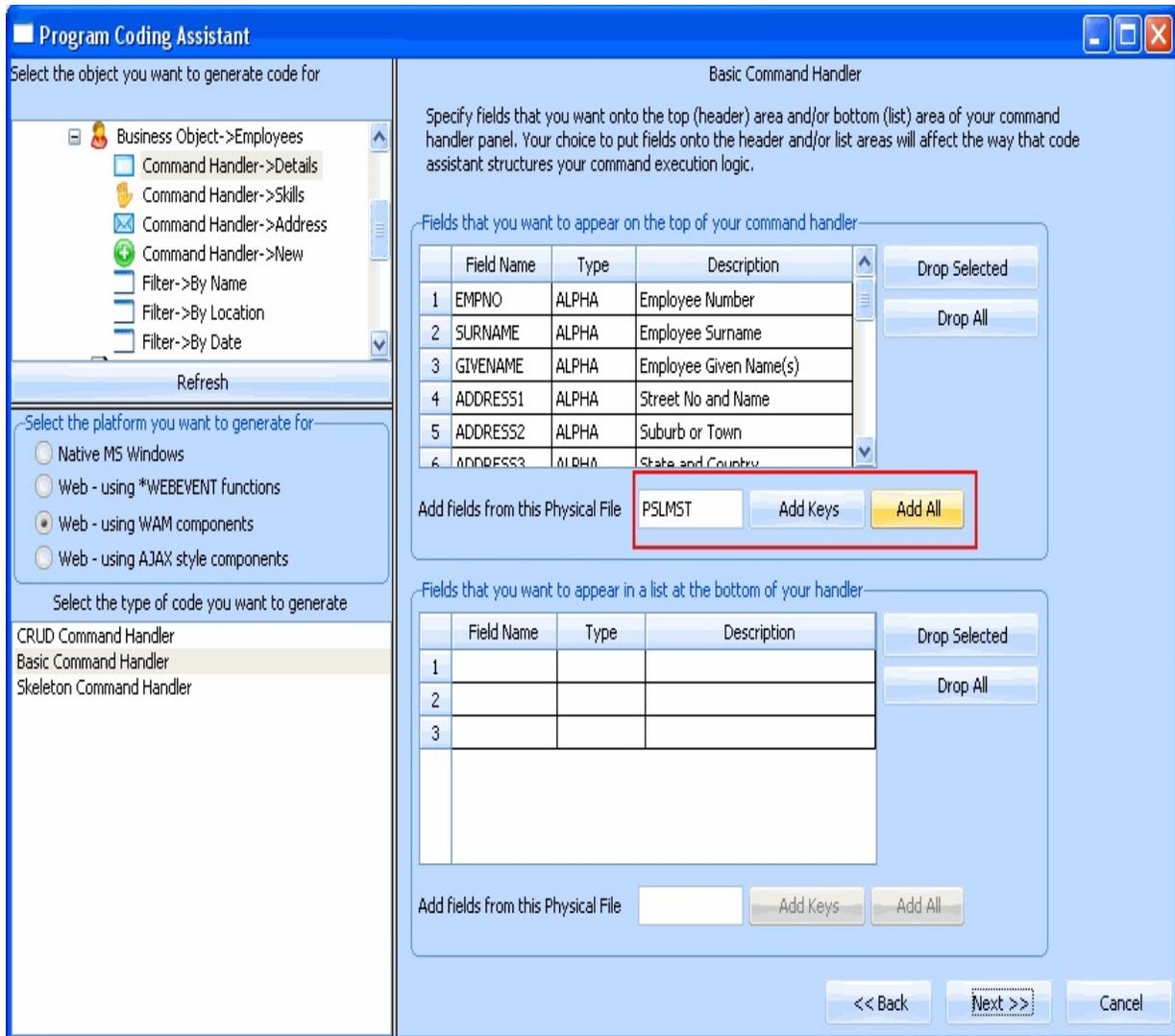
5. Click the Next button.
6. On the next page specify PSLMST as The physical file that most closely

resembles this business object.



The Program Coding Assistant detects the Visual and Programmatic Identifiers required.

7. Click the Next button.
8. On the next page specify PSLMST in the field Add fields from this physical file in the section Fields that you want to appear at the top of your command handler.
9. Click on the Add All button.

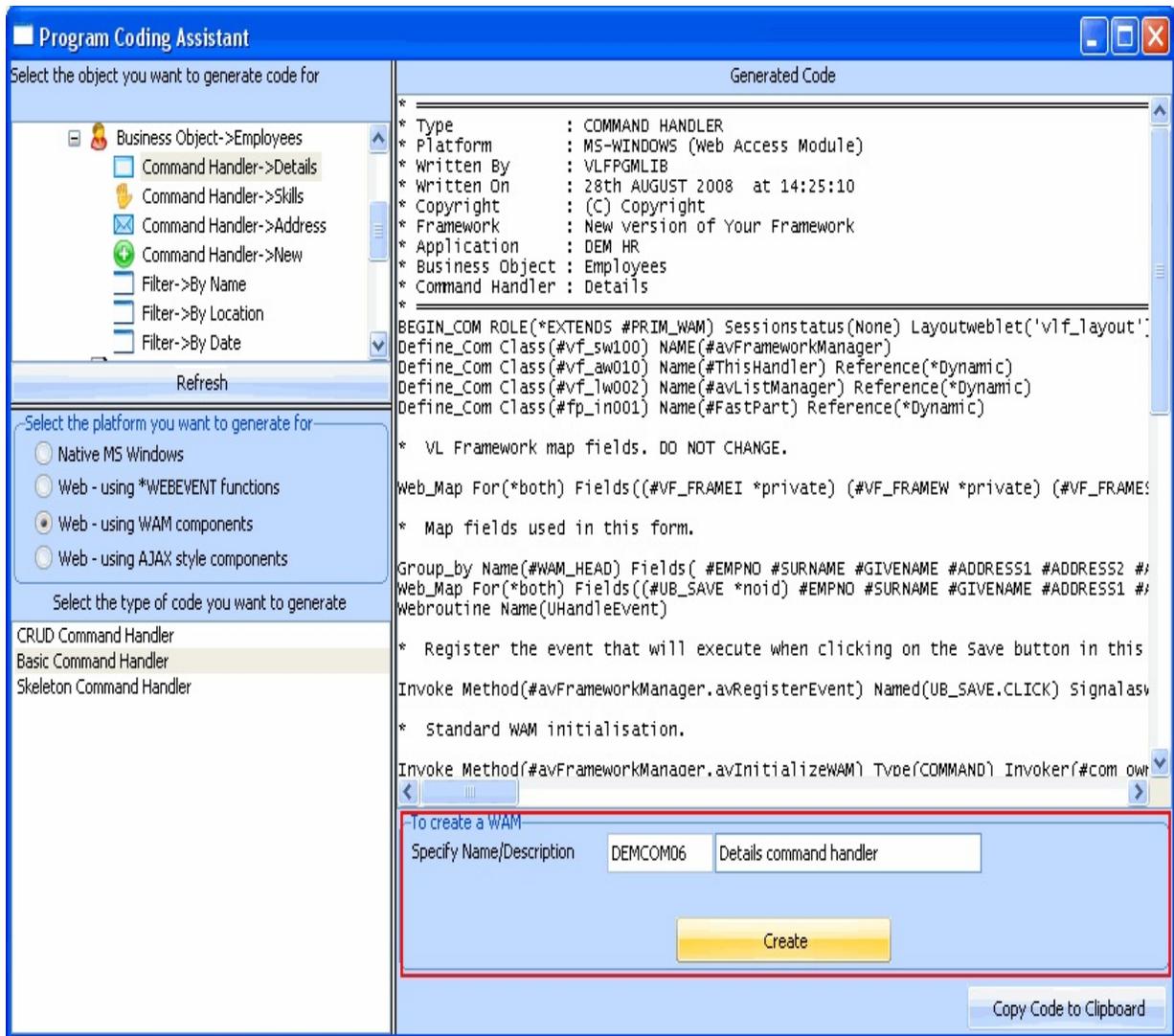


10. On the next page select the Include Default Save Button and Logic and click the Generate Code button.

The next page, Generated Code, displays the source code for your command handler. You now need to create the component that will contain the code:

11. Specify iiiCOM06 (where iii are your initials). Make the description of the component Details command handler.

12. Click on the Create button button and wait until you see a message saying the component has been created in the development environment.



13. Switch to the Visual LANSA editor. The iiiCOM06 WAM is displayed in the editor.

14. If you are using a partition with language NAT, you need to change the default value of the #UB_SAVE field to SAVE and change this Web_Map statement to:

```

Web_Map For(*both) Fields((#UB_SAVE *Desc) #EMPNO #SURNAME
#GIVENAME #ADDRESS1 #ADDRESS2 #ADDRESS3 #POSTCODE
#PHONEHME #PHONEBUS #STARTDTER #TERMDATER
#DEPARTMENT #SECTION #SALARY #STARTDTE #TERMDATE
#MNTHSAL)

```

15. Locate the `#avFrameworkManager.uWAMEvent_1` handler and add a statement to save any changes made to the employee details. For example:

```
UPDATE FIELDS(#WAM_HEAD) IN_FILE(PSLMST)
WITH_KEY(#EMPNO)
```

16. Locate the `uInitialize` event routine. This routine is always called when the command handler is invoked. Notice that it uses `#avListManager.GetCurrentInstance` method to get the key value of the currently selected item.
17. The `uExecute` event routine is only ever executed when the WAM is executed (that is, when a filter is started or a command handler is executed). When events occur inside an active WAM (for example a button click) `uExecute` is not signalled, just the registered `uWAMEvent_N` event.
18. Save the WAM.
19. Check in your changes to the server:

First check in the layout weblet for your WAM::

- a. Locate the layout weblet for your WAM under Weblets in the Repository. It is called `iiiCOM06_layout` (where `iii` correspond to your initials).
- b. Right-click the weblet to bring up the associated pop-up menu and choose the Check in option.
- c. Click OK.

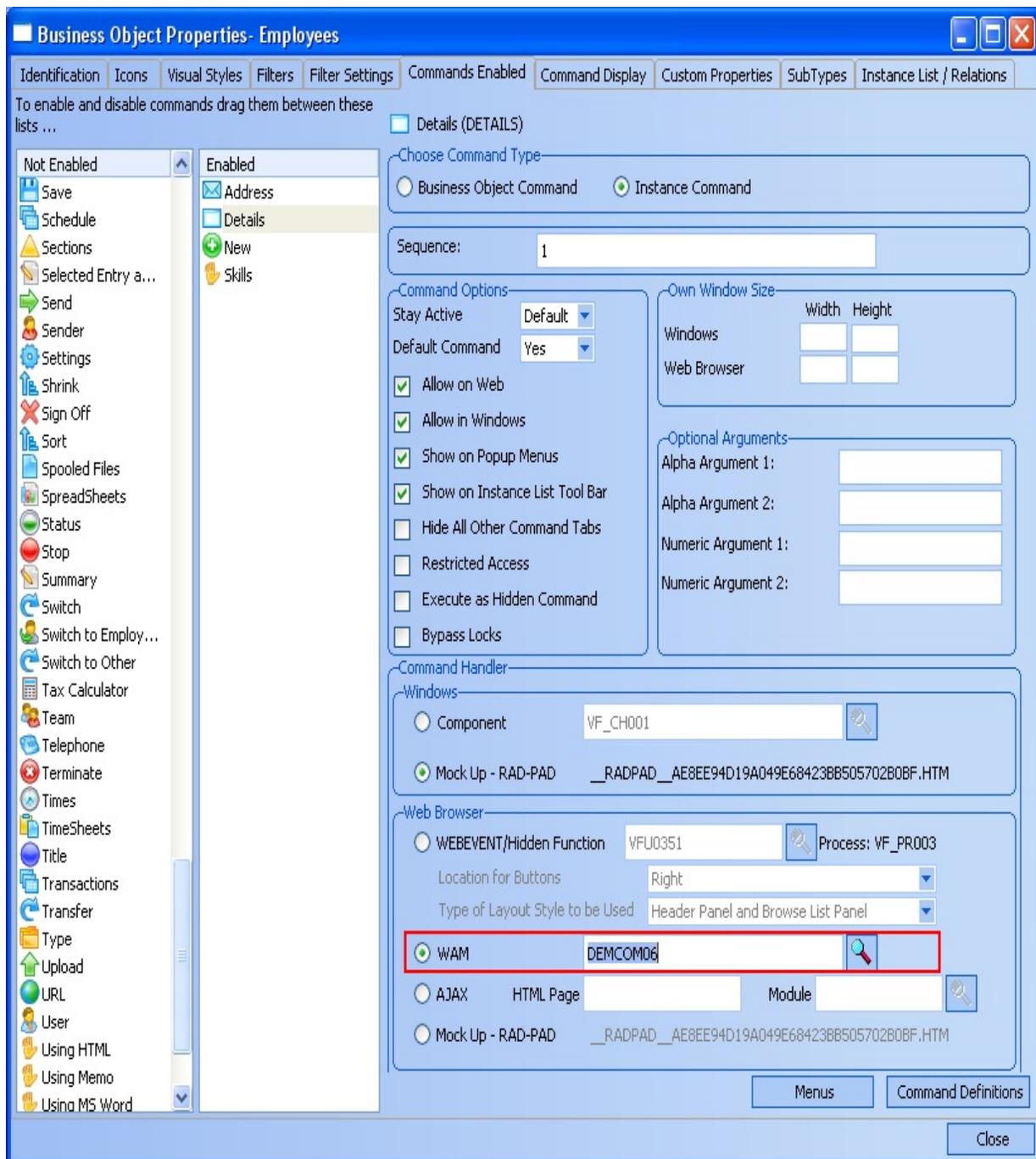
Then:

- a. Compile your WAM locally.
- b. If it compiles ok, select it in the Repository tab.
- c. Right-click the WAM to bring up the associated pop-up menu and choose the Check in option.
- d. In the Check in Options dialog select the option to generate XSL for all webrouines.
- e. Click OK to check the changes in.
- f. Wait until the compiles have finished.

Step 2. Snapping in Your WAM Command Handler

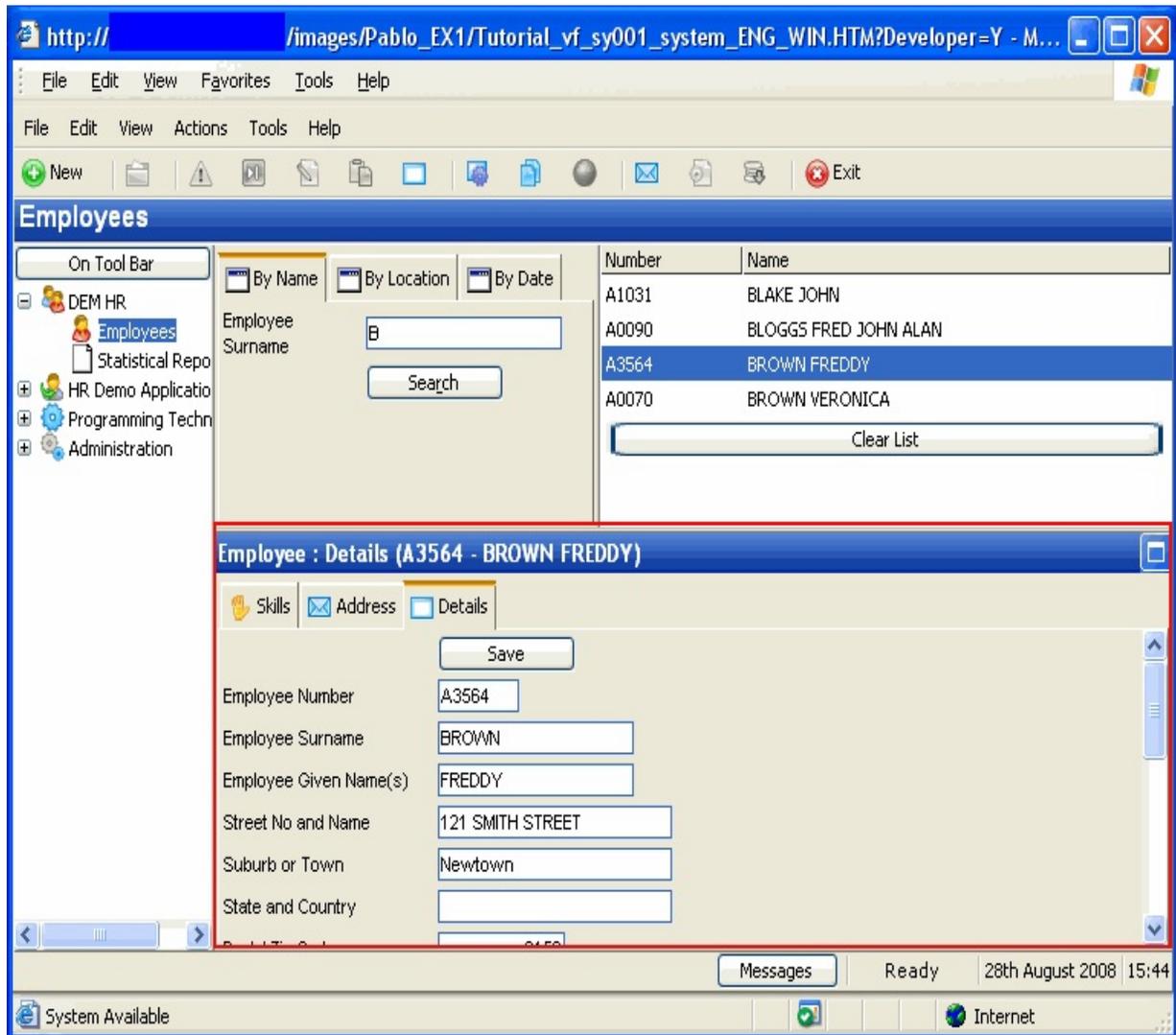
Once you have compiled your command handler and are ready to test it you need to snap it into the Framework. To snap in your own command handler:

1. Display the Framework.
2. Select the iii HR application and display the properties of the Employees object by double-clicking it.
3. On the resulting properties dialog, click on the Commands Enabled tab.
4. Select the Details command.
5. Click on the WAM property radio button in the Web Browser group box.
Type the name of your command handler into the entry field.



6. Use the (Framework) menu and select the option Save the Framework. Accept the prompt to upload the Framework and wait while the upload completes.
7. Use the (Framework) menu and use the option to Execute as Web Application... Select the default options and press OK.

8. Select the iii HR application in the web Framework and then the Employees business object. Bring the By Name filter topmost. Type in a partial surname and click Search. Now click on an employee.
9. Your command handler for Details is now snapped in the web Framework and usable.



Summary

What I Should Know

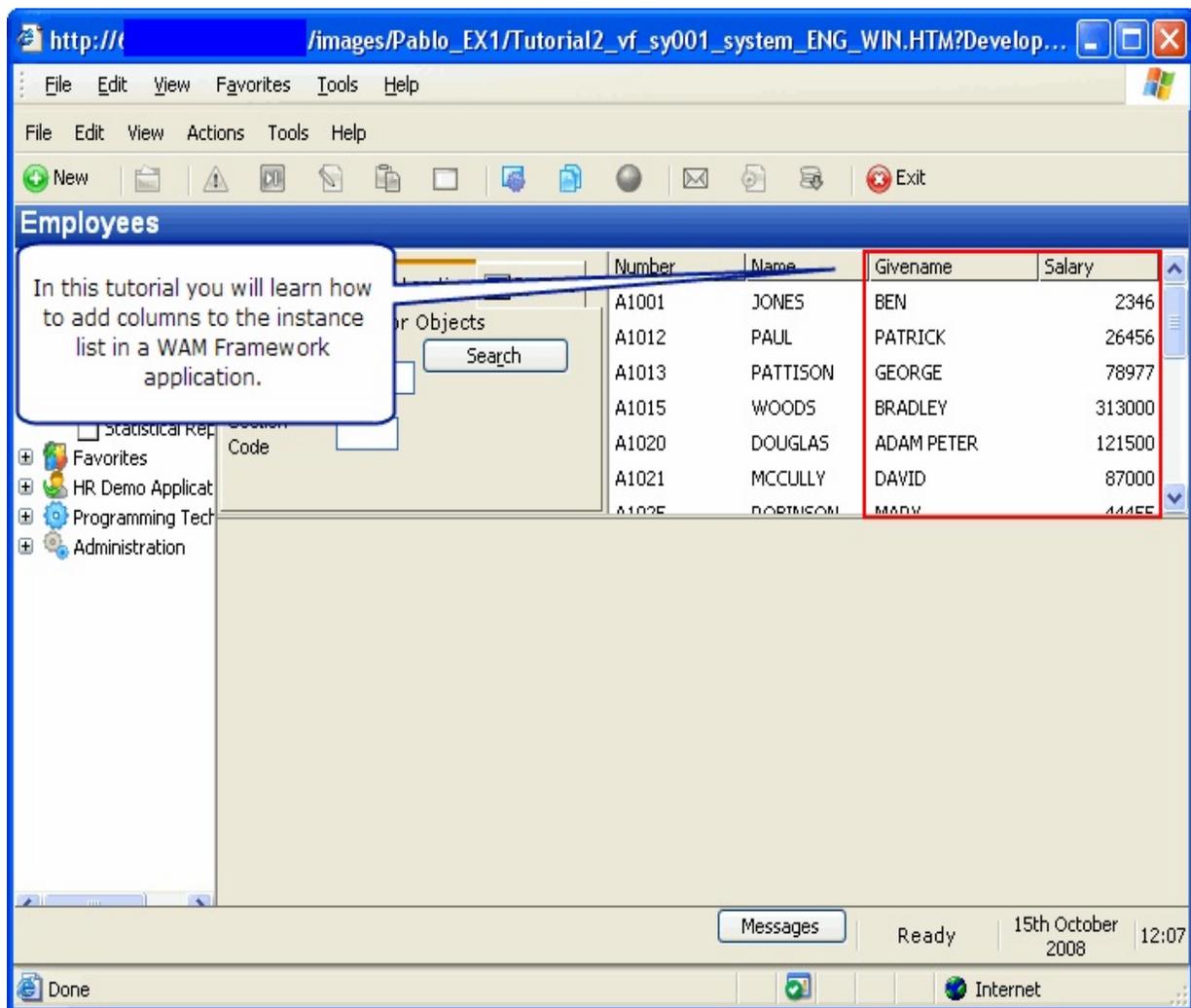
- What you need to do to create your own WAM command handlers.
- How you snap them in the web Framework.

VLF009WAM - Adding Instance List Columns in WAM Applications

Objective

- Learn how to add columns to an [Instance List](#) in a WAM Framework application.

In WAM browser applications, you can add columns to the shipped instance list. Specify the additional columns in the Instance List/Relations tab sheet in the properties of the business object you are working with.



In this tutorial you will learn how to add columns to the instance list in a WAM Framework application.

Number	Name	Givenname	Salary
A1001	JONES	BEN	2346
A1012	PAUL	PATRICK	26456
A1013	PATTISON	GEORGE	78977
A1015	WOODS	BRADLEY	313000
A1020	DOUGLAS	ADAM PETER	121500
A1021	MCCULLY	DAVID	87000
A1025	DORINSON	MARY	44455

Note: in this tutorial, you will modify the By location filter. Normally, you should do the same modifications to the By name filter.

To achieve this objective, you will complete the following steps:

- [Step 1: Add Columns to the Instance List](#)

- [Step 2: Change your filter](#)
- [Step 3: Remove the Additional Columns](#)
- [Summary](#)

Before You Begin

You may wish to review:

- [List Manager](#)
- [Adding Additional Columns to Instance Lists .](#)

In order to complete this tutorial, you must have completed the following:

- [VLF001 - Defining Your HR Application](#)
- [VLF002 - Defining Your Business Objects](#)
- [VLF003 - Prototyping Your Filters](#)
- [VLF004 - Prototyping Your Commands](#)
- [VLF005 - Validating the Prototype](#)
- [VLF006WAM - Snapping in A Real WAM Web Filter](#)
- [VLF007WAM - Snapping in a Real WAM Web Command Handler](#)

Step 1: Add Columns to the Instance List

In this step, you will configure your Employee business object to make the extra columns visible in the instance list.

1. Start the Framework as a designer.
2. Open the properties of the Employees business object.
3. Display the [Instance List/Relations](#) tab sheet.
4. Two visual identifiers are already defined. Add two additional columns:

Column Sequence Column Type Column Caption

30	ACOLUMN1	Givename
40	NCOLUMN1	Salary

Sequence	Type	Caption	Width % (Total 25%)	Decimals	Edit Code	Date/Time Output Format
10	VISUALID1	Number	25		Default	SYSFMT8
20	VISUALID2	Name			Default	SYSFMT8
30	ACOLUMN1	Givename			Default	SYSFMT8
40	NCOLUMN1	Salary			Default	SYSFMT8
	ACOLUMN2				Default	SYSFMT8
	ACOLUMN3				Default	SYSFMT8
	ACOLUMN4				Default	SYSFMT8
	ACOLUMN5				Default	SYSFMT8

Note that the 'Enable Clear List Button' checkbox has no effect in your own Instance Lists.

Step 2: Change your filter

Next, you need to make changes to your filter to fill the extra fields in the instance list with data.

1. Open the By Location filter iiiCOM05 which you created in [VLF006WAM - Snapping in A Real WAM Web Filter](#).
2. Make these changes to the code:
 - a. Change the GROUP_BY command to include the #SALARY field:

```
Group_By Name(#XG_Ident) Fields( #EMPNO #SURNAME  
#GIVENAME #SALARY)
```

- b. Locate Select Fields(#XG_Ident) command and change the AddtoList statement to:

```
* Add instance details to the instance list  
#avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#Surname)  
AKey1(#EMPNO) AColumn1(#Givename) NColumn1(#Salary)
```

3. Save the changed source code.
4. If your Web server is on a System i, use the Host Monitor to send your changes to the System i.
5. Compile your new WAM on the server.
6. Restart the Framework and test the result.

http:// /images/Pablo_EX1/Tutorial2_vf_sy001_system_ENG_WIN.HTM?Develop...

File Edit View Favorites Tools Help

File Edit View Actions Tools Help

New [Icons] Exit

Employees

On Tool Bar

- DEM HR
 - Company Dep
 - Department S
 - Employees**
 - Statistical Rep
- Favorites
- HR Demo Applicat
- Programming Tech
- Administration

By Name **By Location** By Date

Search for Objects

Department Code

Section Code

Number	Name	Givenname	Salary
A1001	JONES	BEN	2346
A1012	PAUL	PATRICK	26456
A1013	PATTISON	GEORGE	78977
A1015	WOODS	BRADLEY	313000
A1020	DOUGLAS	ADAM PETER	121500
A1021	MCCULLY	DAVID	87000
A1025	DORINSON	MARY	11155

Messages Ready 15th October 2008 12:07

Done [Icons] Internet

Step 3: Remove the Additional Columns

In this step you will remove the additional columns.

1. Display the properties of the Employees business object.
2. Display the Instance List / Relations tab.
3. Remove the column sequence numbers for Givenname and Salary fields.
4. Close the properties and save the Framework.

Summary

What You Should Know

- How to add columns to a instance list in a browser WAM Framework application.

VLF011WAM - Creating a Parent Child Instance List

Objectives

- To learn how to create a parent-child instance list (see [Instance Lists with Different Types of Objects](#)) with a [Hidden Filter](#) and a relationship handler.

In this tutorial you will create a parent-child instance list.

The list will be populated by a hidden filter and a relationship handler.

Company Department	Description	Street	Town/Suburb	State and Country	Post Code	Phone
ADM	ADMINISTRATOR DEPT				0	
Department Sections						
INTERNAL ADMIN SRV	01	125 Main St,	Blacktown	NSW	2167	679 2536
PURCHASING SECTION	02	123 Pacific Highway,	North Sydney, 2000	NSW	2000	952 6475
ACCOUNTING SECTION	03	252 Canterbury Road,	CANTERBURY, NSW.		2044	560 3633
SALES & MARKETING	04	121 Pitt Town Road	Pitt Town	Northern Region	2345	364-8905
MAINTENANCE	05	121 Railway Parade	Woodsville		2034	456-(02)

System Available | Messages | Ready | 6th January 2009 14:54 | Internet

To achieve this objective, you will complete the following steps:

Step 1. [Create Two New Business Objects](#)

Step 2. [Establish the Parent-Child Relationship](#)

Step 3. [Create a Hidden Filter for Company Departments](#)

Step 4. [Create a Relationship Handler to Load Sections](#)

Step 5. [Display Additional Columns in the Instance List](#)

Step 6. [Access the Properties of Hidden Child Objects](#)

Summary

Before You Begin

In order to complete this tutorial, you must have completed the following:

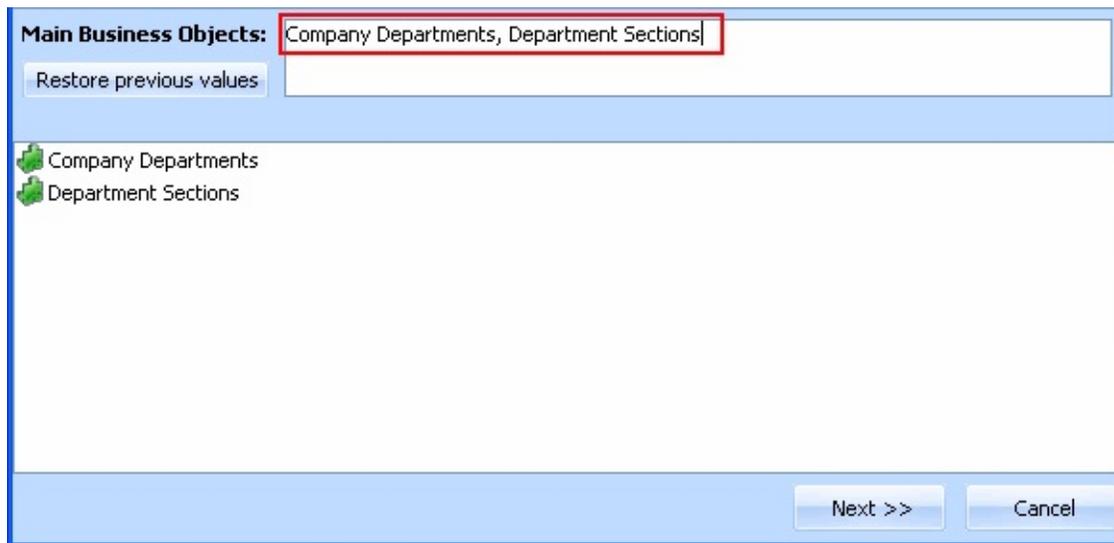
- Tutorials VLF000 – VLF007WAM.

Step 1. Create Two New Business Objects

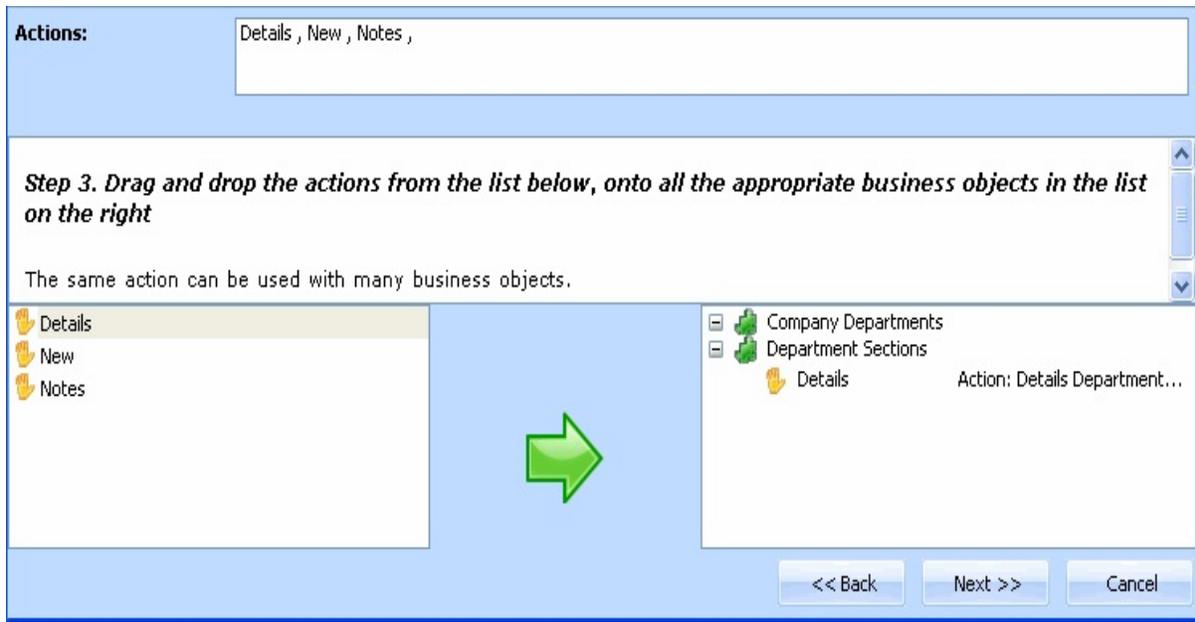
In this step you will use the Instant Prototyping Assistant to create two business objects: Company Departments and Department Sections.

The Sections business object will become the child of the Company Departments object.

1. Start the Instant Prototyping Assistant in the Framework.
2. Type in the names of two new business objects Company Departments and Department Sections. Remember to separate the names with a comma.

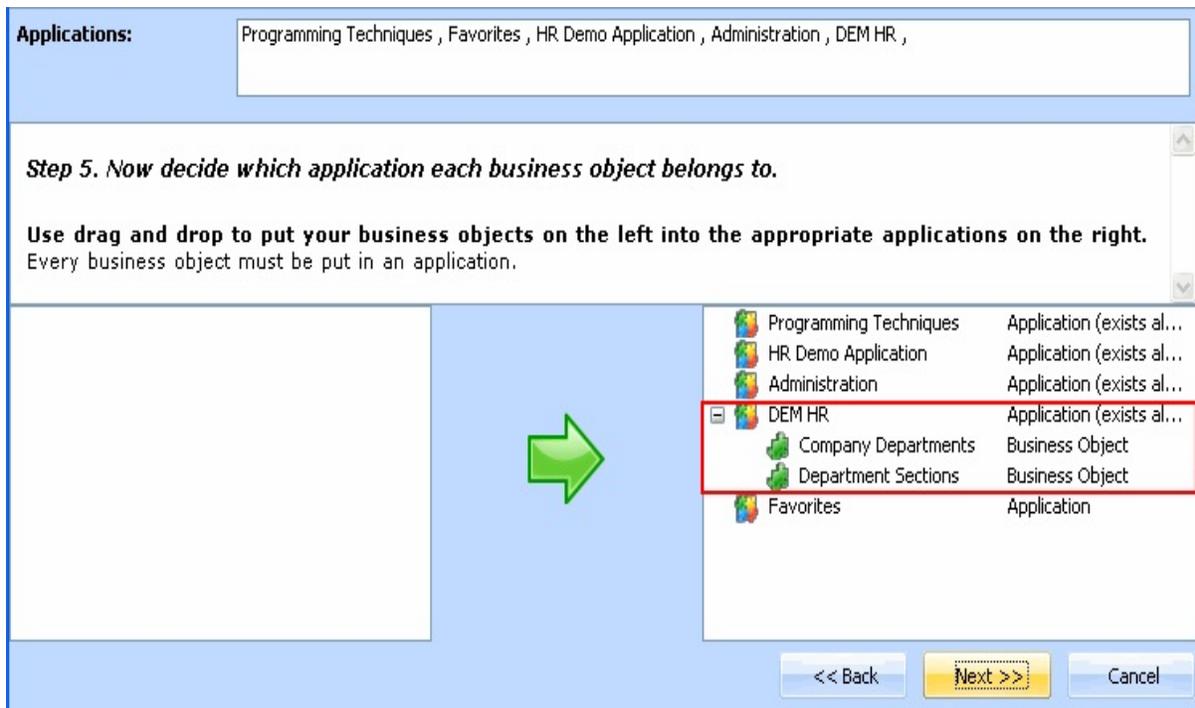


3. Click Next.
4. Drag the Details command to the Department Sections business object.



5. Click Next.

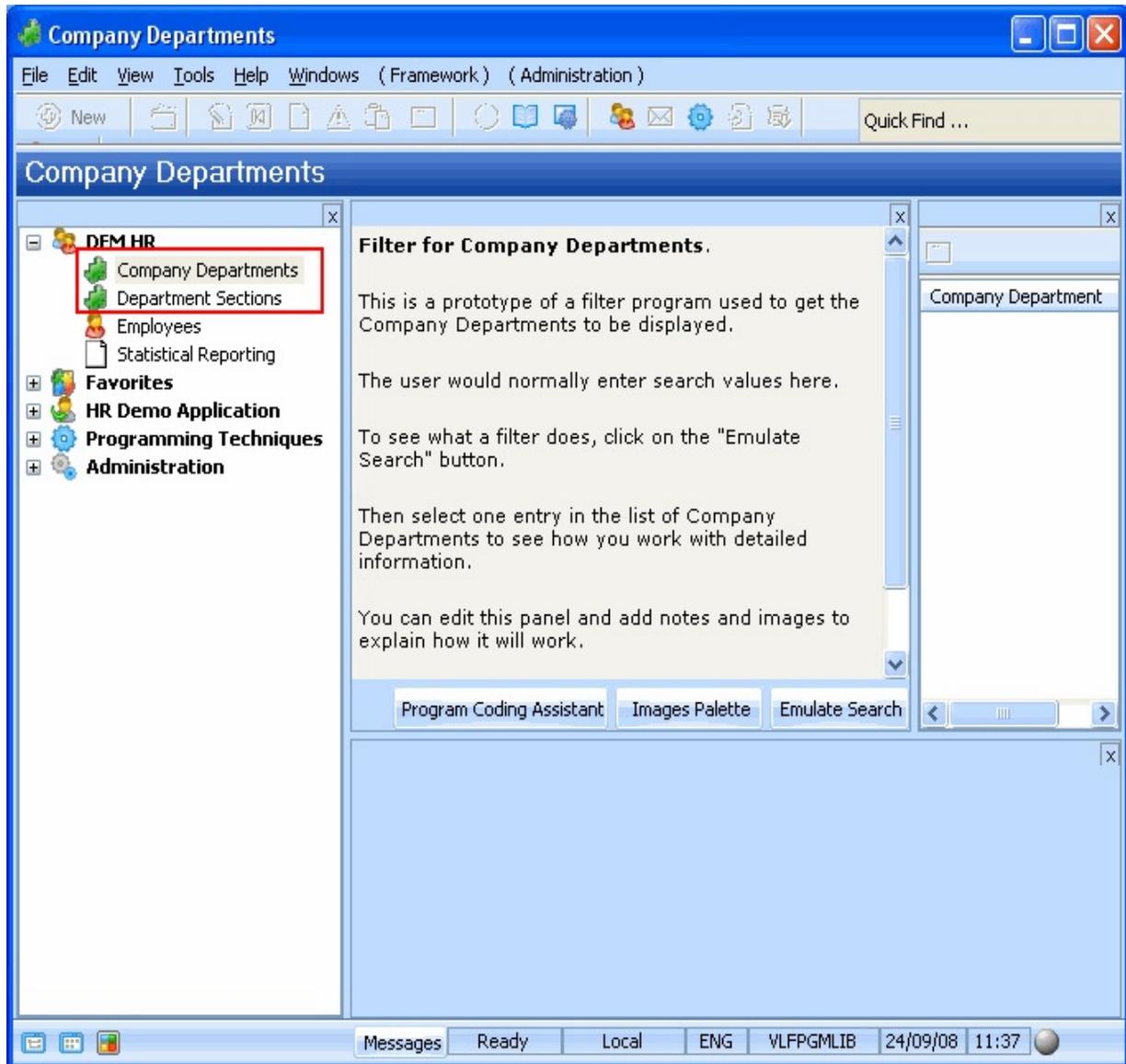
6. Drag both business objects to the iii HR application.



7. Click Next.

8. Click Finish.

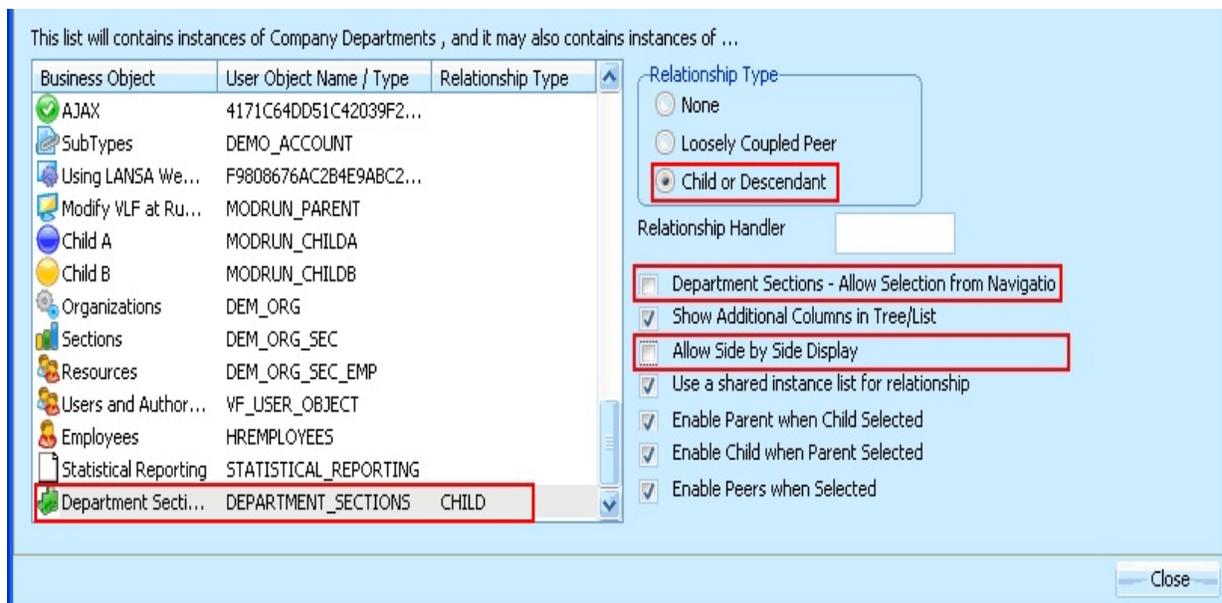
The new business objects are now visible in the iii HR application:



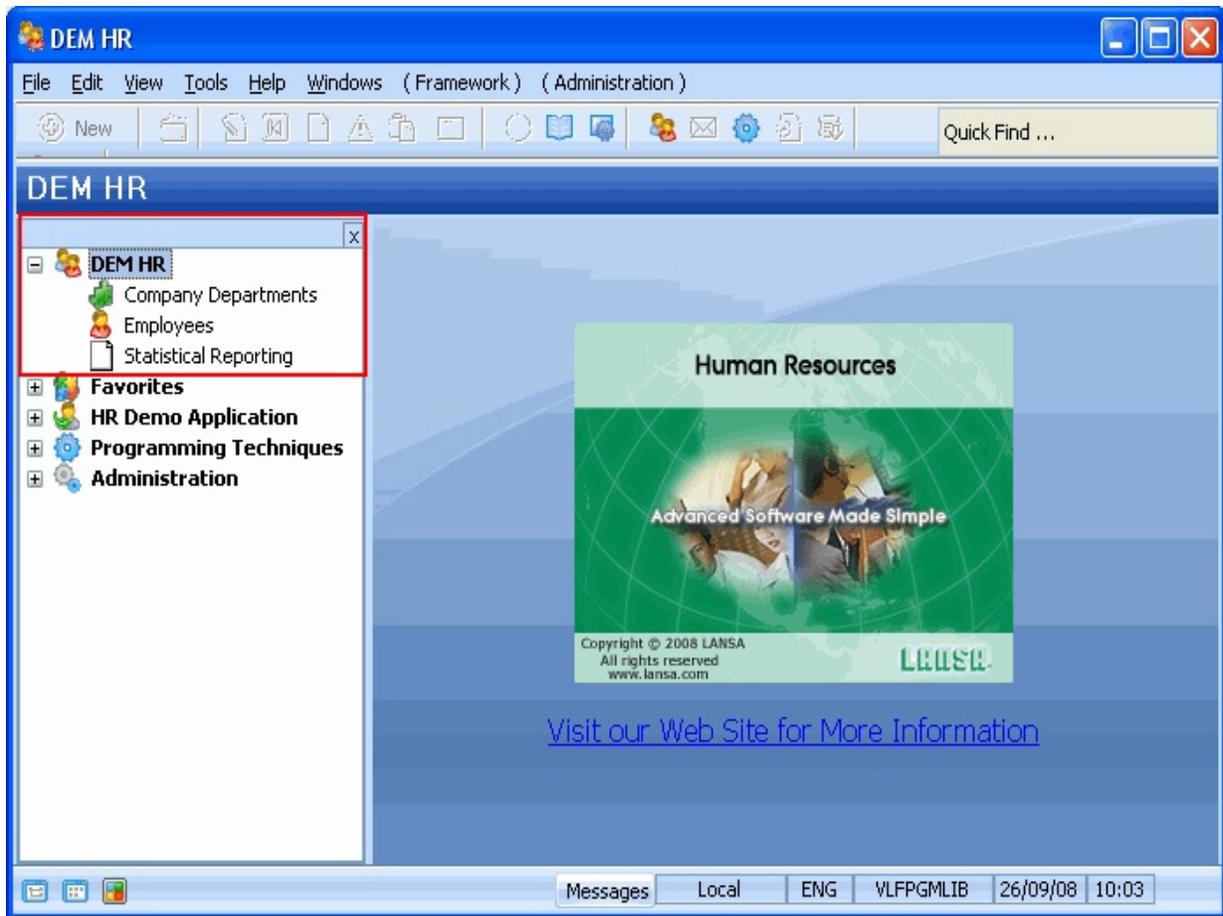
Step 2. Establish the Parent-Child Relationship

In this step you will establish the relationship between the Company Departments and Department Sections business objects.

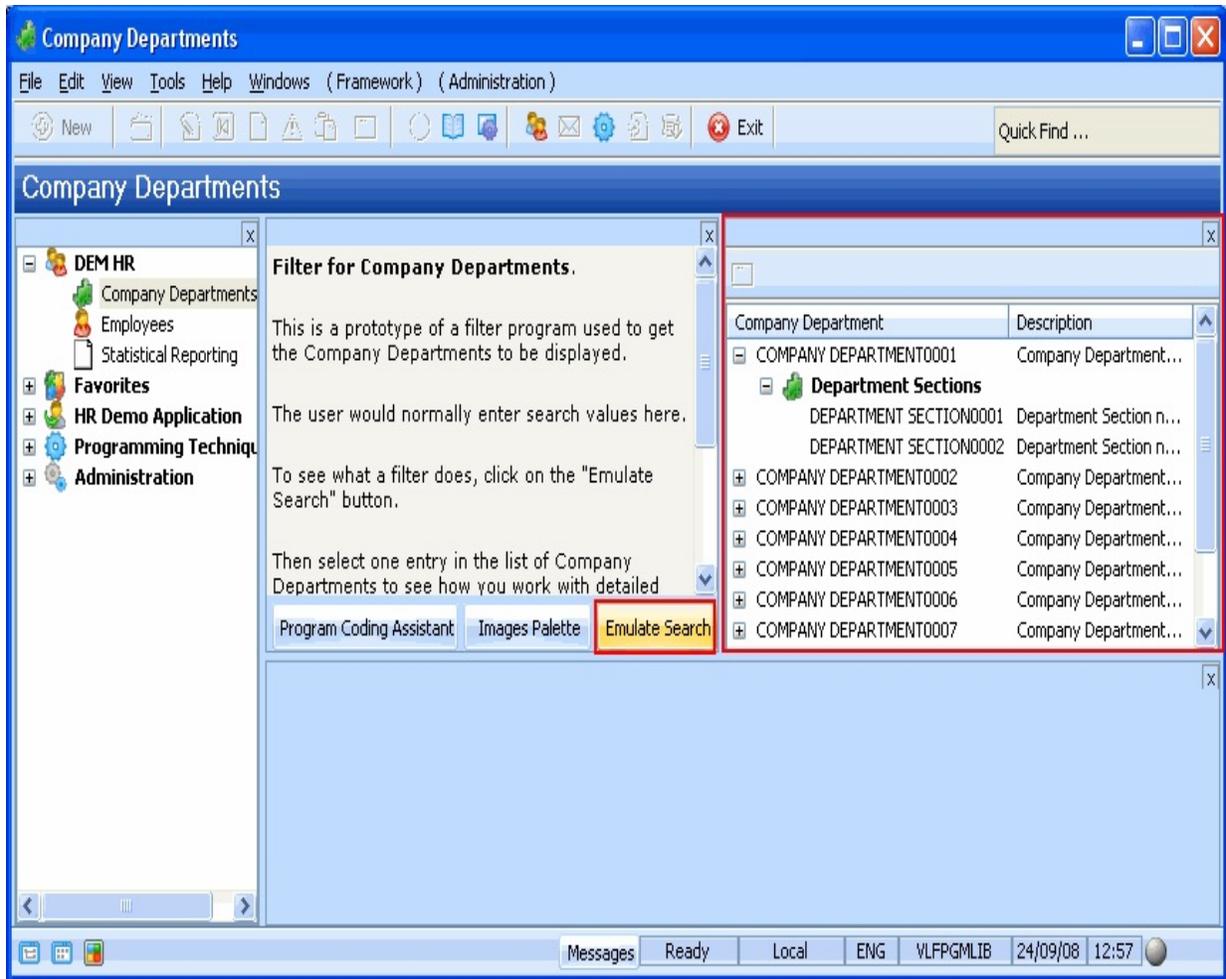
1. Display the properties of the Company Departments object.
2. Click on the Instance List / Relations tab.
3. In the list in the lower left portion of the screen scroll down and select the Department Sections business object.
4. In the Relationship Type group box on the right select the Child or Descendant radio button.
5. Deselect the check box Allow Selection from Navigation Pane.
6. Deselect the check box Allow Side by Side display.
8. Click the Close button in the message that appears.



9. Click the Close button.
10. Save and restart the Framework.
11. Open the iii HR application. Notice that the Department Sections business object is no longer displayed in the navigation pane.



12. Select the Company Departments business object and click on the Emulate Search button in the mock-up filter. Expand one of the Company Departments in the instance list. Notice that the emulated data shows its child business objects, the Department Sections.



13. Close the application.

Step 3. Create a Hidden Filter for Company Departments

In this step you will create a hidden filter that loads the Company Departments to the instance list when you select 1. the Company Departments business object in the navigation pane. With a hidden filter there is no end-user interaction and the filter is not visible.

1. Start the Framework.
2. Start the Program Coding Assistant.
3. Select the iiiHR application and then the Company Departments business object, then New Filter.
4. Select Web – Using WAM components as the platform.
5. Select A skeleton filter as the type of component you want to create.

The screenshot shows the 'Program Coding Assistant' dialog box. The left pane is titled 'Select the object you want to generate code for' and contains a tree view with the following items: Application->Administration, Application->DEM HR (highlighted with a red box), Business Object->Employees, Business Object->Statistical Reporting, Business Object->Company Department, Filter->New Filter, Business Object->Department Sections, and Command Handler->About... Below the tree is a 'Refresh' button. The right pane is titled 'Select the platform you want to generate for' and contains three radio buttons: 'Native MS Windows', 'Web - using *WEBEVENT functions', 'Web - using WAM components' (selected and highlighted with a red box), and 'Web - using AJAX style components'. Below this is the section 'Select the type of code you want to generate' with the following options: 'CRUD Filter', 'Filter that searches using a file or view', 'A skeleton filter' (highlighted with a red box), 'Search button event handling routine (code fragment)', and 'Invoke #avListManager.AddtoList (code fragment)'. The right pane also contains a 'Generate Code' button. Below the button is a 'Generated Code' window showing the following code:

```
Generated Code
Type : BUSINESS OBJECT FILTER
Platform : MS-WINDOWS (Visual LANG)
Architecture : XP_X64
Version : 1.0.0.0
Created On : 10/11/2008 10:11:28
Copyright : IBM Corporation
Product : IBM WebSphere
Application : Data Application
Business Object : Employee
Filter : By Name
=====
FUNCTION OPTIONS/FORMATT
Name: Filter (Business Object Filter)
Description: Search and selection definitions
=====
SEARCH BY : NAME(S)
DEF_LIST : NAME(S)
DEF_COND : NAME(S)
=====
Component Definitions
```

The right pane also contains the following text:

What? This assistant produces the skeleton code for a filter. The code produced requires manual completion. It is designed for use by developers who have created filters before and only want a skeleton filter to manually add their own specialized code to.

Who? Experienced Developers

Where? Browser platforms (generates WAM RDML code)

How? Click the *Generate Code* button to generate the outline filter code.

Enter a name and description and click the *Create* button to directly create a new WAM from the generated code, or, Click the *Copy Code*

Buttons: Generate Code, Cancel

It is recommended that you use the Program Code Assistant when creating WAM filters or command handlers to make sure they meet all the Framework rules.

6. Click Generate Code.
7. In the Generated Code page specify iiiCOM11 as the name of your filter and Departments Hidden Filter as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
8. Click Create.

After a brief delay a message is shown indicating the WAM has been created.

9. Switch to the Visual LANSA editor to see your filter skeleton.
10. Locate the event routine handling the #avFrameworkManager.uexecute event.
11. In the event routine add a statement to indicate that the filter will be hidden:

```
Set Com(#thisfilter) Avhiddenfilter(true)
```

12. Add these statements to start updating the instance list and then clear it:

```
Invoke Method(#avListManager.BeginListUpdate)  
Invoke Method(#avListManager.ClearList)
```

13. Select Departments and their descriptions from the DEPTAB file:

```
Select Fields(#DEPARTMENT #DEPTDESC) From_File(DEPTAB)  
With_key(#DEPARTMENT ) Generic(*yes)  
  
Invoke #avListManager.AddtoList Visualid1(#DEPARTMENT)  
Visualid2(#DEPTDESC) AKey1(#DEPARTMENT)  
  
EndSelect
```

14. Lastly indicate that the instance list update is complete:

Invoke Method(#avListManager.EndListUpdate)

Your code will now look like this:

```
Evtroutine Handling(#avFrameworkManager.uexecute) Options(*noclearmessages *noclearerrors)
  Set Com(#thisfilter) Avhiddenfilter(true)

  Invoke Method(#avListManager.BeginListUpdate)
  Invoke Method(#avListManager.ClearList)

  Select Fields(#DEPARTMENT #DEPTDESC) From_File(DEPTAB) With_key(#DEPARTMENT ) Generic(*yes)
    Invoke #avListManager.AddtoList Visualid1(#DEPARTMENT) Visualid2(#DEPTDESC) AKey1(#DEPARTMENT)
  EndSelect

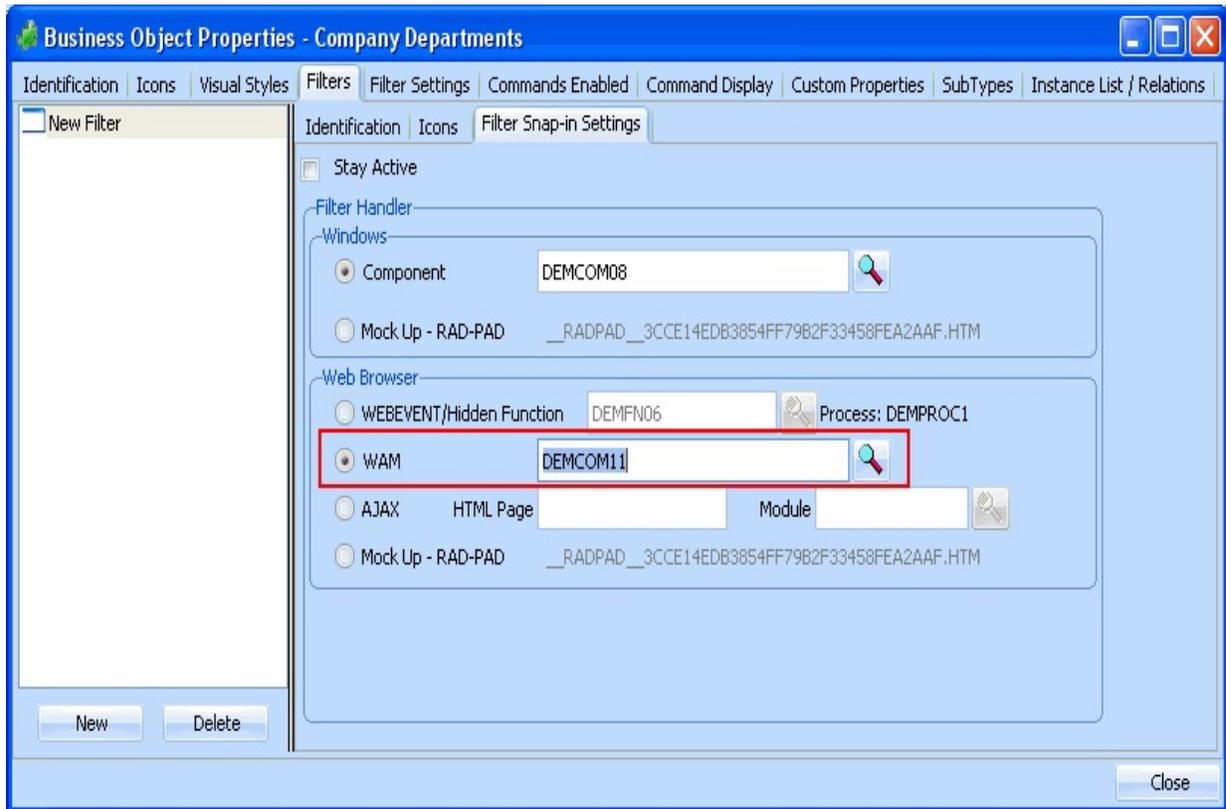
  Invoke Method(#avListManager.EndListUpdate)
Endroutine
```

15. Select the Compile option in the Verify menu.
16. Ensure that the All webroutines option is selected for the Generate XSL option.
17. Ensure the compilation was successful.
18. Check in your changes to the server:
 - a. Right-click the WAM to bring up the associated pop-up menu and choose the Check in option.
 - b. Ensure that the All webroutines option is selected for the Generate XSL option.
 - c. Click OK to check the changes in.
 - d. Wait until the compiles have finished.

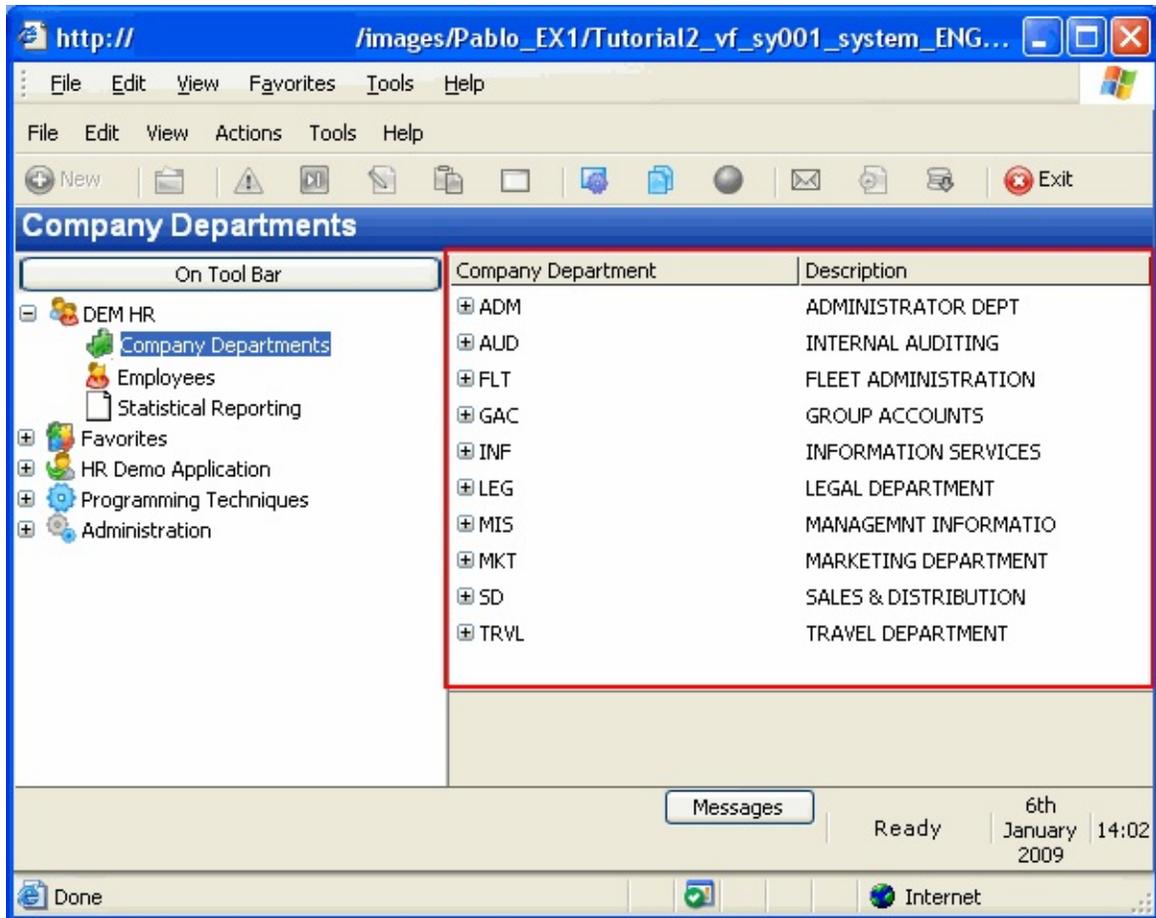
Now that you have compiled your new filter and are ready to test it, you need to snap it into the Framework.

19. In the Framework, close the Program Coding Assistant.
20. Select the iii HR application and double-click on the Company Departments business object.
21. On the resulting Business Object Properties dialog, click on the Filters tab.

22. Select the New Filter which is created by default.
23. Click on the Filter Snap-in Settings Tab.
24. Click on the WAM property radio button in the Web Browser group box.
25. Type the name of your WAM filter (iiiCOM11) into the entry field.



26. Close the Company Departments' properties.
27. Save the Framework. Accept the prompt to upload the Framework and wait while the upload completes.
28. Use the (Framework) menu and select the option to Execute as Web Application... Accept the default options and press OK.
29. Select the iii HR application in the web Framework and then the Company Departments business object



30. Expand a Department. Notice that no Department Sections are loaded. You will create the relationship handler that loads the sections in the next step.
31. Close the application.

Step 4. Create a Relationship Handler to Load Sections

In this step you will create a relationship handler that loads Sections into the instance list when a Department is expanded.

You could have loaded all the Sections in the hidden filter code together with the Departments, but by using a relationship handler you can improve filter performance by first only adding root or parent objects to the instance list and then dynamically adding the child objects.

1. In the Visual LANSA editor, create a process iiiPROC2 – Framework Functions. Create a function belonging to this process. Specify iiiFN04 as the name of your function and Relationship Handler as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
2. Replace the existing code in the function with this code that indicates that this function is a relationship handler:

```
FUNCTION OPTIONS(*DIRECT *LIGHTUSAGE) RCV_LIST(#VIS_LIST  
#PID_LIST #COL1_LIST #COL2_LIST #COL3_LIST #COL4_LIST  
#COL5_LIST #COL6_LIST #COL7_LIST #COL8_LIST #COL9_LIST  
#COLA_LIST)
```

```
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL1)  
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL2)
```

The VFREL1 and VFREL2 functions which you include contain the standard definitions for relationship builder functions.

3. Next clear all the keys and additional columns in the instance list:

```
EXECUTE SUBROUTINE(CLEARKEYS)  
EXECUTE SUBROUTINE(CLEARCOLS)
```

The subroutines you call in the relationship handler are contained in the VFREL2 function.

4. Get the key value of the selected department:

```
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)
```

5. Select the sections in the current department and set the values of the instance list entry:

```
SELECT FIELDS(*ALL) FROM_FILE(SECTAB)
WITH_KEY(#DEPARTMENT)
  EXECUTE SUBROUTINE(SETAKEY) WITH_PARM(1 #DEPARTMENT)
  EXECUTE SUBROUTINE(SETAKEY) WITH_PARM(2 #SECTION)
  EXECUTE SUBROUTINE(SETNCOL) WITH_PARM(1 #SECPCODE)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(1 #SECADDR1)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(2 #SECADDR2)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(3 #SECADDR3)
  EXECUTE SUBROUTINE(SETACOL) WITH_PARM(4 #SECPHBUS)
  EXECUTE SUBROUTINE(ADDTOLIST)
WITH_PARM('DEPARTMENT_SECTIONS' #SECDESC #SECTION)
ENDSELECT
```

- The SETAKEY subroutine sets the key values of the child instance list. The first parameter of the subroutine is the key position and the second parameter is the value of the key. There is also a SETNKEY subroutine to set a numeric key.
- The SETNCOL and SETACOL subroutines add additional columns for the child instance list entry.
- The ADDTOLIST subroutine adds the entry to the instance list. The first parameter of the subroutine is the child business object name, the second parameter is the Visual ID 1 column and the third parameter is the Visual ID 2 column.

Your code will now look like this:

```

FUNCTION OPTIONS(*DIRECT *LIGHTUSAGE)
  RCV_LIST(#VIS_LIST #PID_LIST #COL1_LIST #COL2_LIST #COL3_LIST #COL4_LIST #COL5_LIST
    #COL6_LIST #COL7_LIST #COL8_LIST #COL9_LIST #COLA_LIST)

☒ INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL1)
☒ INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL2)

* Clear all keys and additional columns
EXECUTE SUBROUTINE(CLEARKEYS)
EXECUTE SUBROUTINE(CLEARCOLS)

* Return the sections in a department/organization
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)

* Read all the Sections in the specified
* department and add them to the instance list

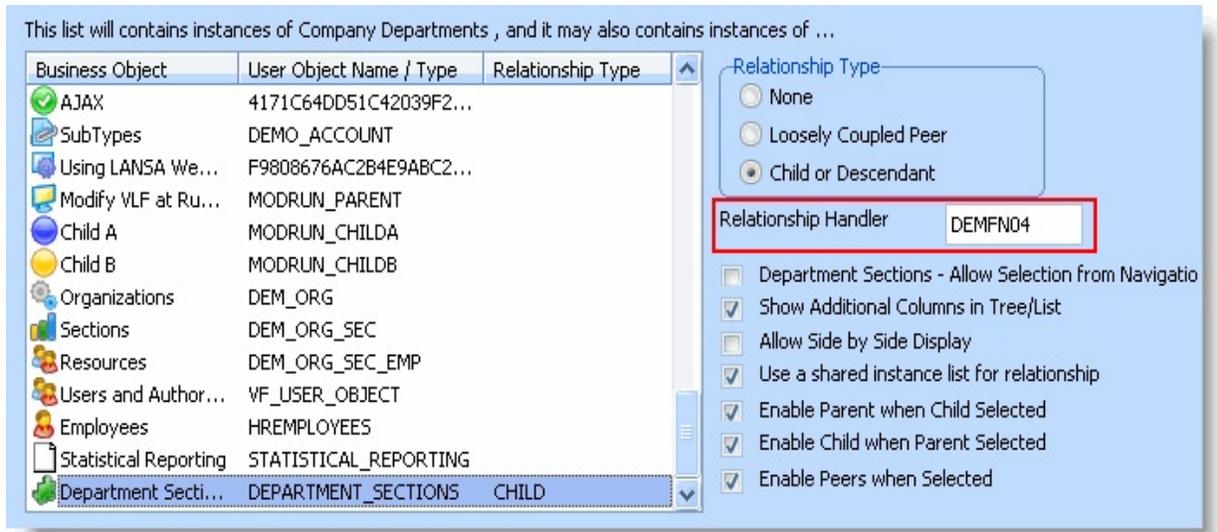
☒-SELECT FIELDS(*ALL) FROM_FILE(SECTAB) WITH_KEY(#DEPARTMENT)
DCM0123/Warning : no key specified to match file SECTAB from DC@DEMOLIB key field SECTION

EXECUTE SUBROUTINE(SETAKEY) WITH_PARS(1 #DEPARTMENT)
EXECUTE SUBROUTINE(SETAKEY) WITH_PARS(2 #SECTION)
EXECUTE SUBROUTINE(SETNCOL) WITH_PARS(1 #SECPCODE)
EXECUTE SUBROUTINE(SETACOL) WITH_PARS(1 #SECADDR1)
EXECUTE SUBROUTINE(SETACOL) WITH_PARS(2 #SECADDR2)
EXECUTE SUBROUTINE(SETACOL) WITH_PARS(3 #SECADDR3)
EXECUTE SUBROUTINE(SETACOL) WITH_PARS(4 #SECPHBUS)
EXECUTE SUBROUTINE(ADDTOLIST) WITH_PARS('DEPARTMENT_SECTIONS' #SECDESC #SECTION)

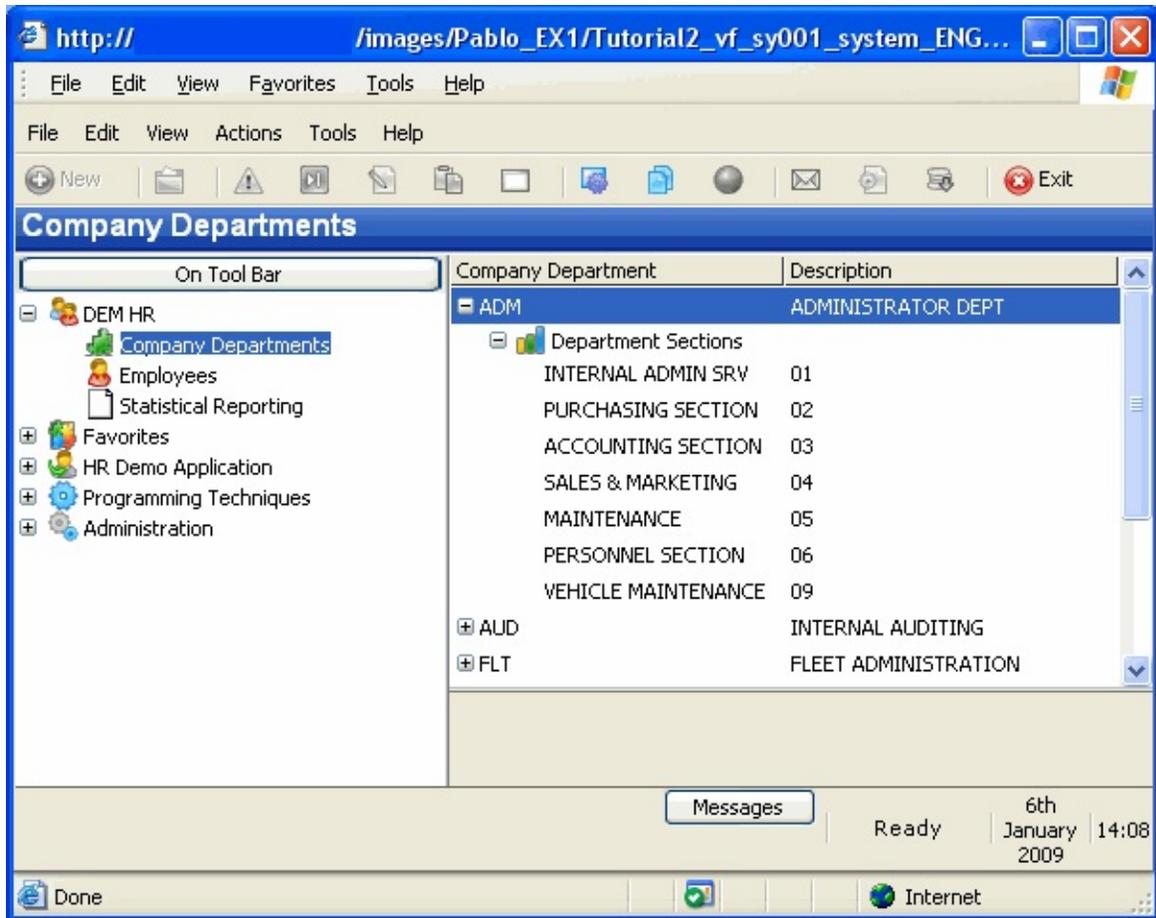
ENDSELECT

```

6. Compile the function.
7. Check in your changes to the server:
 - a. Right-click the WAM to bring up the associated pop-up menu and choose the Check in option.
 - b. Ensure that the All webroutines option is selected for the Generate XSL option.
 - c. Click OK to check the changes in.
 - d. Wait until the compiles have finished.
8. Display the Framework.
9. Display the properties of the Company Departments business object.
10. In the Instance List/Relations tab select the Department Sections business object.
11. In the Relationship Handler field, type in the name of the relationship handler.



12. Close the Company Departments properties.
13. Save the Framework. Accept the prompt to upload the Framework and wait while the upload completes.
14. Use the (Framework) menu and select the option to Execute as Web Application... Accept the default options and press OK.
15. Select the iii HR application in the web Framework and then the Company Departments business object
16. Expand a department in the instance list and then the sections underneath it.



The sections in each department you expand are loaded dynamically.

Note that only the section name and identifier are shown in the list. In the next step you change the instance list to show additional columns for the sections.

17. Close the application.

Step 5. Display Additional Columns in the Instance List

In this step you create additional columns in the instance list to show all the data loaded for sections by the relationship handler.

1. Display the properties of the Company Departments business object.
2. In the Instance List / Relationships tab specify these additional columns:

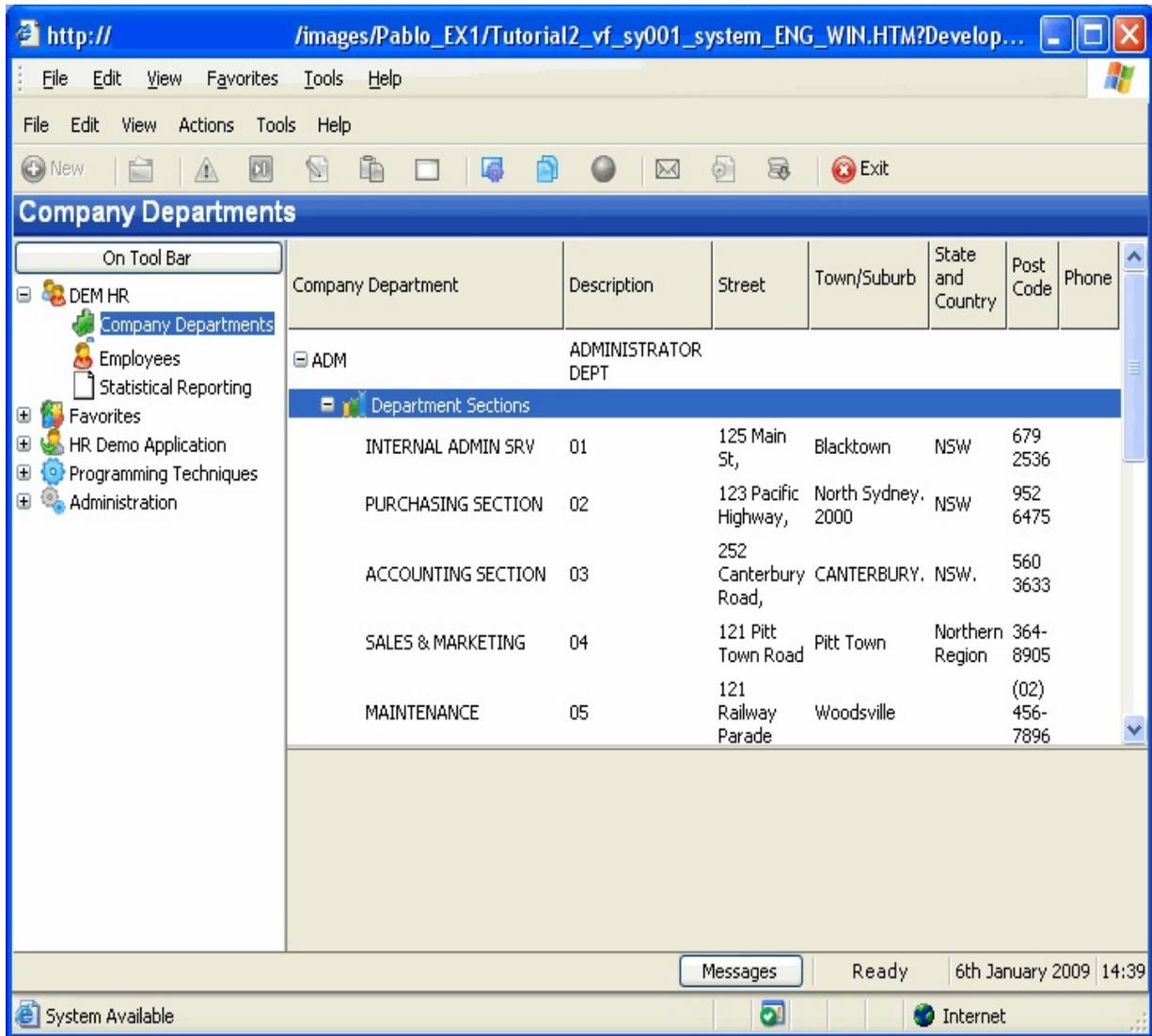
Column Sequence Column Type Column Caption

30	ACOLUMN1	Street
40	ACOLUMN2	Town/Suburb
50	ACOLUMN3	State and Country
60	NCOLUMN1	Post Code
70	ACOLUMN4	Phone

Your instance list column definitions now look like this:

Sequence	Type	Caption	Width % (Total 25%)
10	VISUALID1	Company Department	25
20	VISUALID2	Description	
30	ACOLUMN1	Street	
40	ACOLUMN2	Town/Suburb	
50	ACOLUMN3	State and Country	
60	NCOLUMN1	Post Code	
70	ACOLUMN4	Phone	
	ACOLUMN5		

3. Close the properties and save the Framework. Accept the prompt to upload the Framework and wait while the upload completes.
4. Use the (Framework) menu and select the option to Execute as Web Application...
Accept the default options and press OK.
5. Select the iii HR application in the web Framework and then the Company Departments business object
6. Expand a department in the instance list and then the sections underneath it.

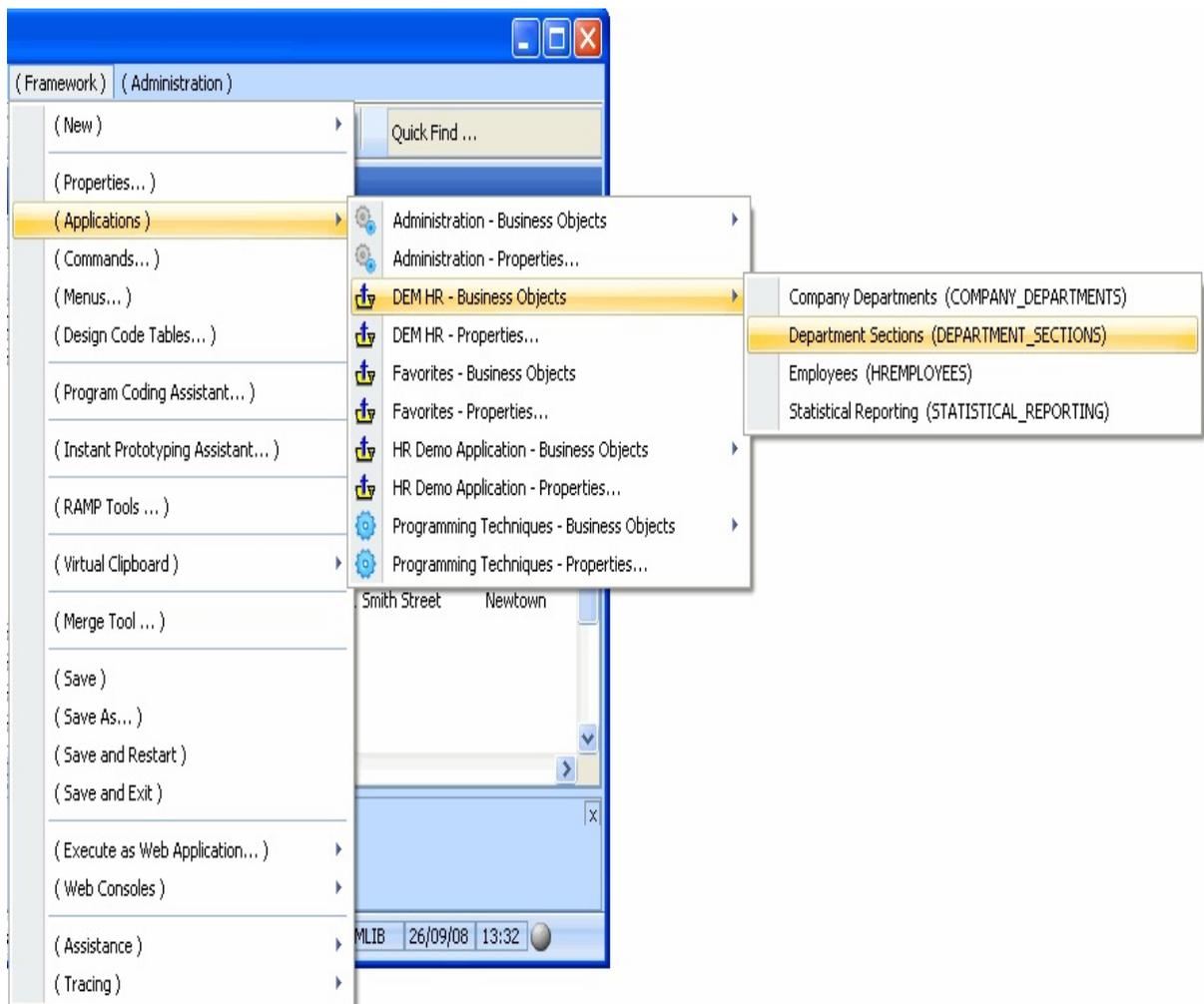


7. Close the application.

Step 6. Access the Properties of Hidden Child Objects

In this step you will learn how to access the properties of the hidden child business object Department Sections which is not visible in the navigation pane.

1. Display the Framework menu and select the Applications... menu option.
2. Select the iii HR application.
3. Select the Department Sections business object to display the properties of the Department Sections business object.



4. Close the properties of the Department Sections business object.

There is also an alternative way of displaying the properties of child business

objects which are not accessible from the navigation pane:

5. Display the sections in a department in the instance list.
6. Double-click on a section to display the properties of the Department Sections business object.

Summary

Important Observations

- You can create instance lists that contain more than one type of object. You do this by defining relationships between business objects. The relationships can either be peer-to-peer or parent-child.
- In situations where you want to completely fill the business object instance list programmatically, the filter has no meaningful interaction with the end-user and can be hidden from view.
- A relationship handler is an RDML function that is called to dynamically expand the relationship between a parent and child object. By doing this you can improve filter performance by only adding root or parent objects to the instance list initially.
- The Framework instance list can display up to 10 alphanumeric and/or 10 numeric additional columns in an instance list.

Tips & Techniques

- The Advanced section of the Programming Techniques sample application has examples of advanced instance lists.
- LANSAs supplies a sample relationship handler to copy from when creating your relationships. The source is stored in function DF_REL01 in the process DF_PROC.

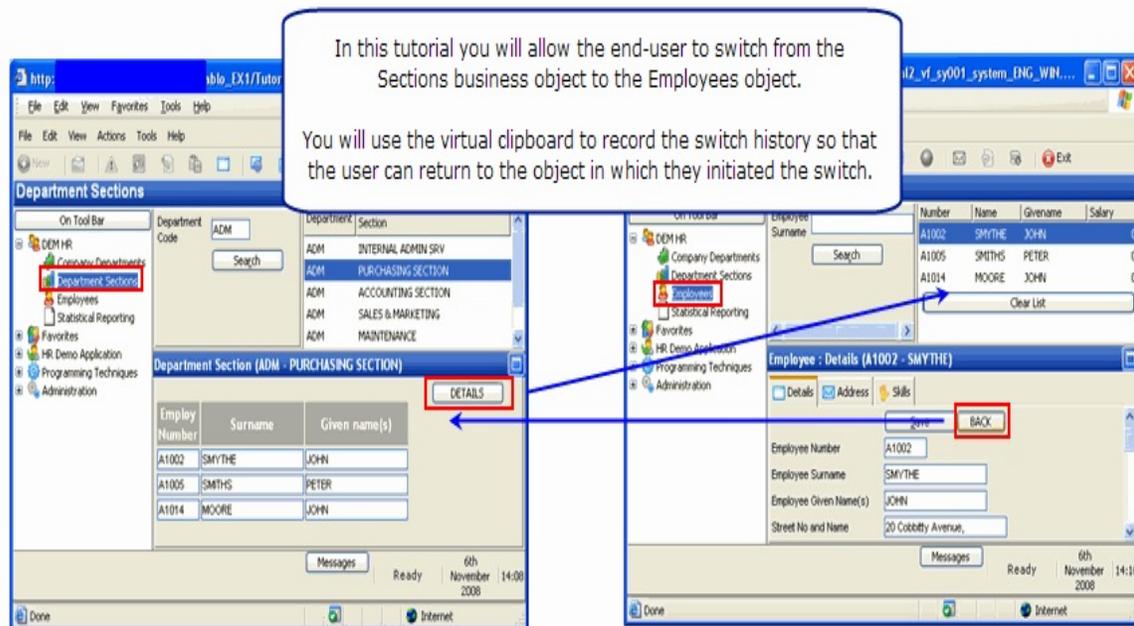
What I Should Know

- How to create a parent-child relationship between business object
- How to create a hidden filter
- How to write a relationship handler
- How to add additional columns to the instance list

VLF012WAM - Controlling Navigation Using Switching and the Virtual Clipboard

Objectives

- To learn how to use switching to swap control between different business objects and to execute commands at the Framework, application or business object level (see [Object Switching Service](#)).
- To learn to use [The Virtual Clipboard](#) to store the switch history.



To achieve this objective, you will complete the following steps:

Step 1. Create a Filter for Department Sections

Step 2. Create a Details Command Handler for Department Sections

Step 3. Add Logic to Switch from Sections to the Employees Business Object

Step 4. Record Switch History using the Virtual Clipboard

Step 5. Use the Switch History to Return to the Original Business Object

Summary

Before You Begin

In order to complete this tutorial, you must have completed the following:

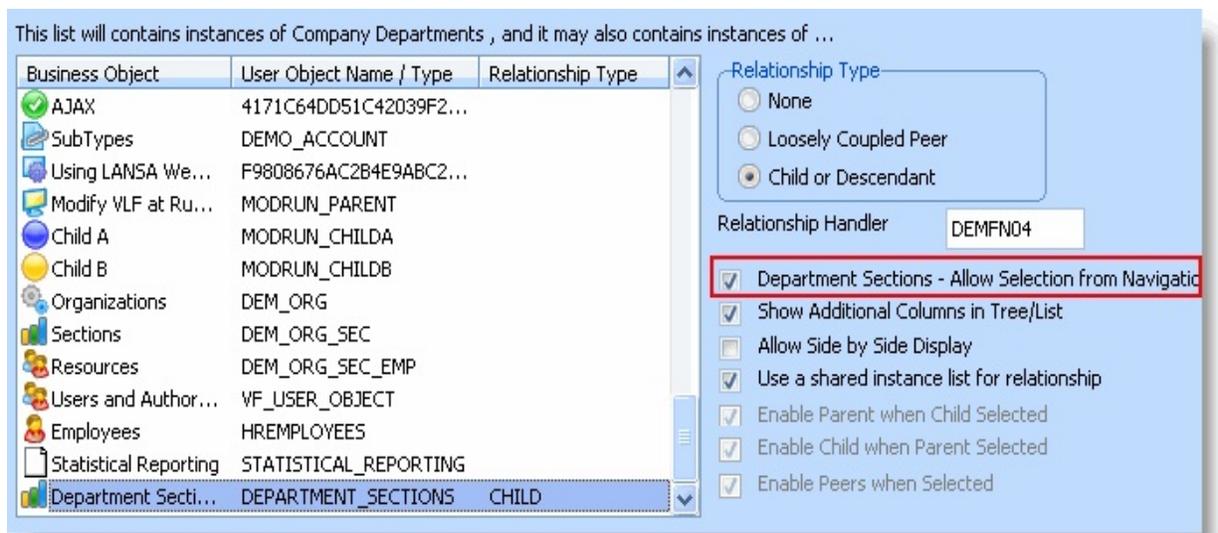
- Tutorials VLF000 – VLF007WAM and VLF011WAM

Step 1. Create a Filter for Department Sections

In this step you will make the Department Sections business object visible in the navigation pane and create a filter for it.

You need to do this in preparation for the switching exercise because object switching can only be performed on objects which are visible in the navigation pane.

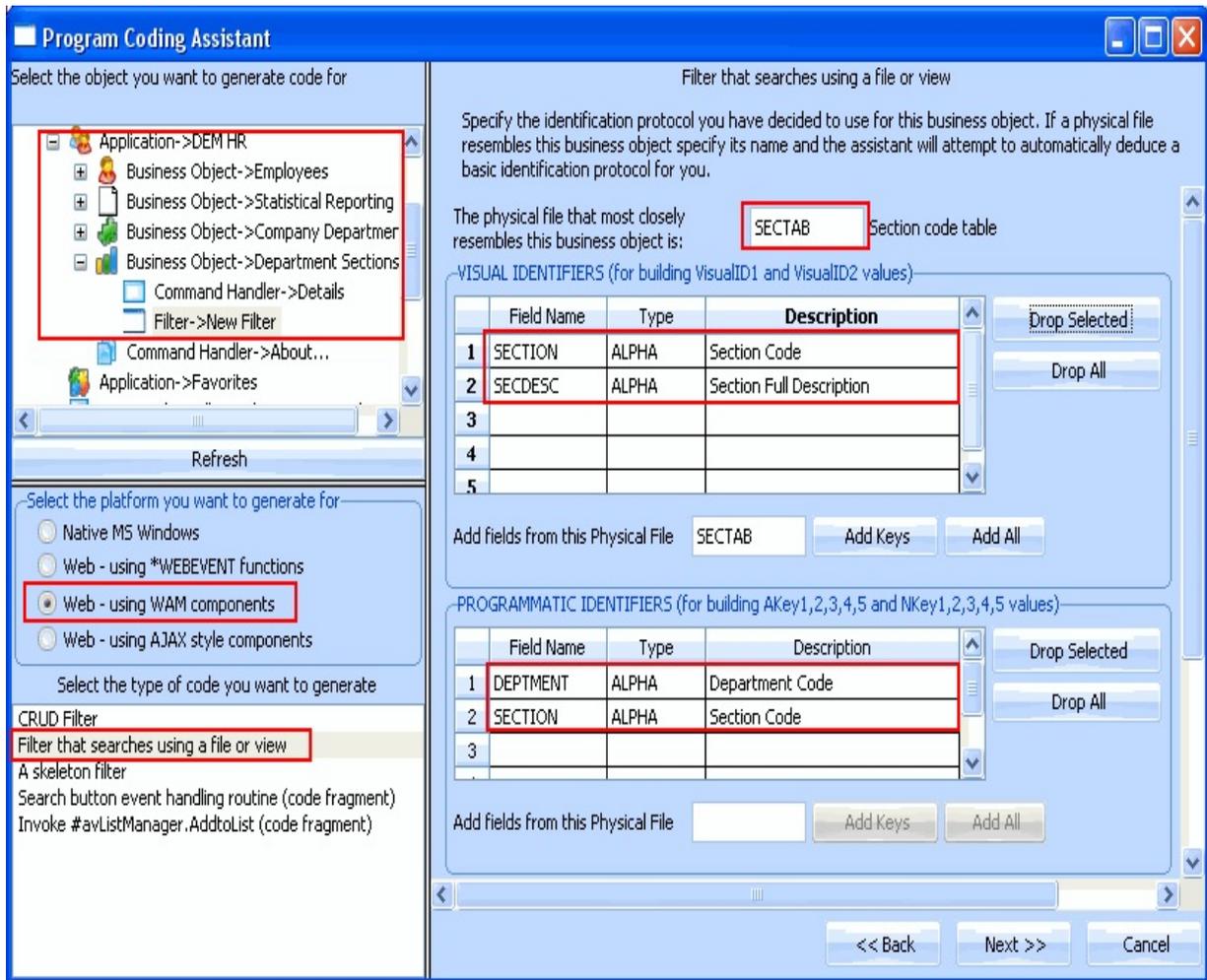
1. In the Framework, display the properties of the Company Departments business object.
2. Display the Instance List / Relationships tab.
3. Select Department Sections in the list on the bottom left.
4. Select the option Department Sections – Allow Selection from Navigation Pane.



5. Close the properties of the Company Departments business object. The Department Sections business object is now visible in the navigation pane.
6. Display the properties of the Department Sections business object.
7. Change the icon for example to 🏢.

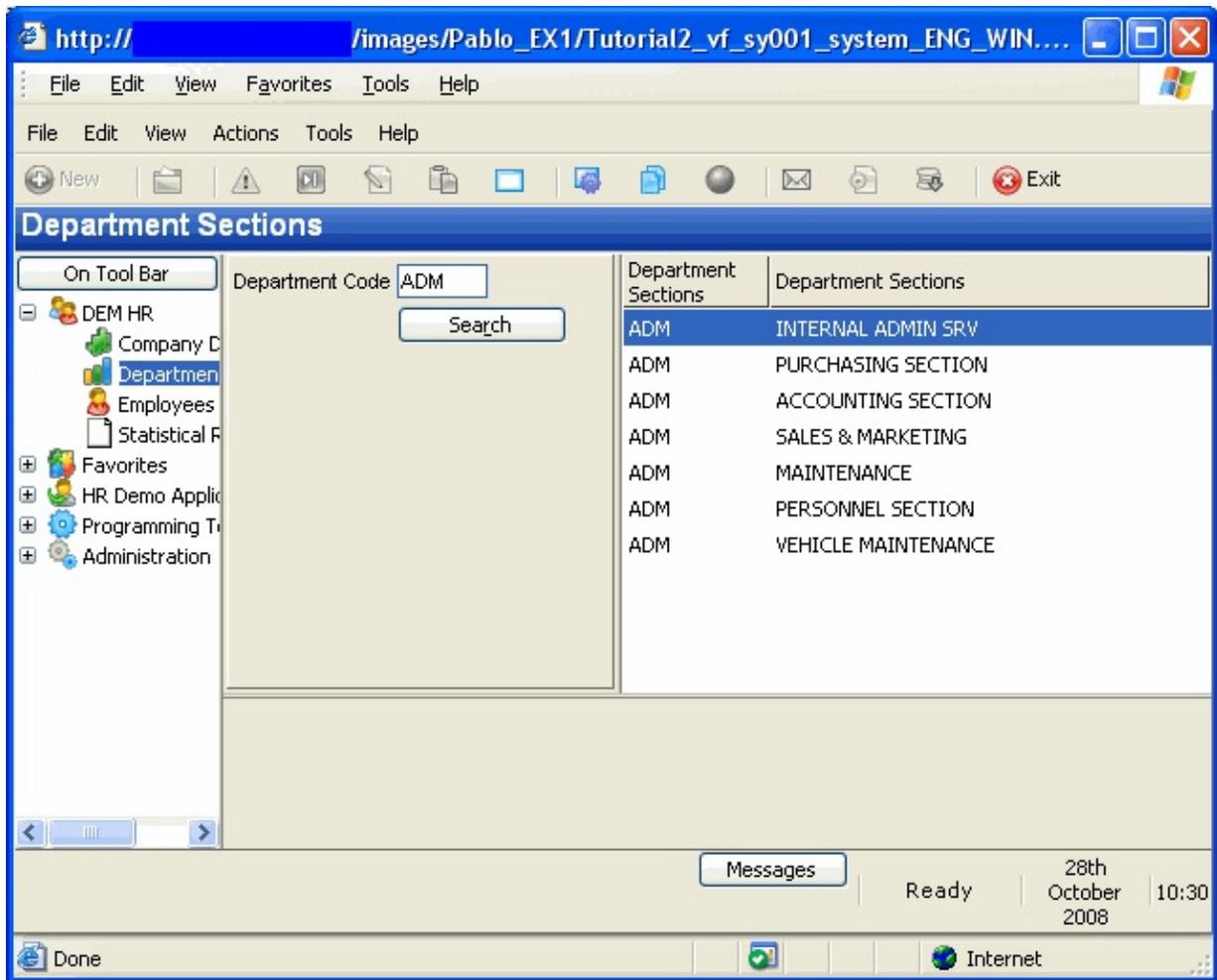
Next you need to replace the mock-up filter in the Sections business object with a functional filter to populate the instance list:

8. Start the Program Coding Assistant.
9. Select the Department Sections business object in the iii HR application.
10. Select New Filter, WAM as the platform and a Filter that searches using a file or a view.
11. Click Next.
12. Specify SECTAB as the physical file, and SECTION and SECDESC as the visual identifiers.



13. Accept the other defaults set by the Program Coding Assistant and click Next.
14. Specify DEPTMENT field as the key to be used for search operations.
15. Click Generate Code.

16. On the Generated Code page specify iiiCOM12 as the name of your filter and Sections Filter as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
17. Click Create. You will shortly see a message saying the component has been created.
18. Switch to the Visual LANSA Editor.
19. Compile the filter.
20. Check the filter and its associated layout weblet into the server.
21. In the Framework, snap the filter in the Department Sections business object.
22. Save the Framework definition and upload it to the server.
23. Execute the Framework as a web application.
24. Test the filter.

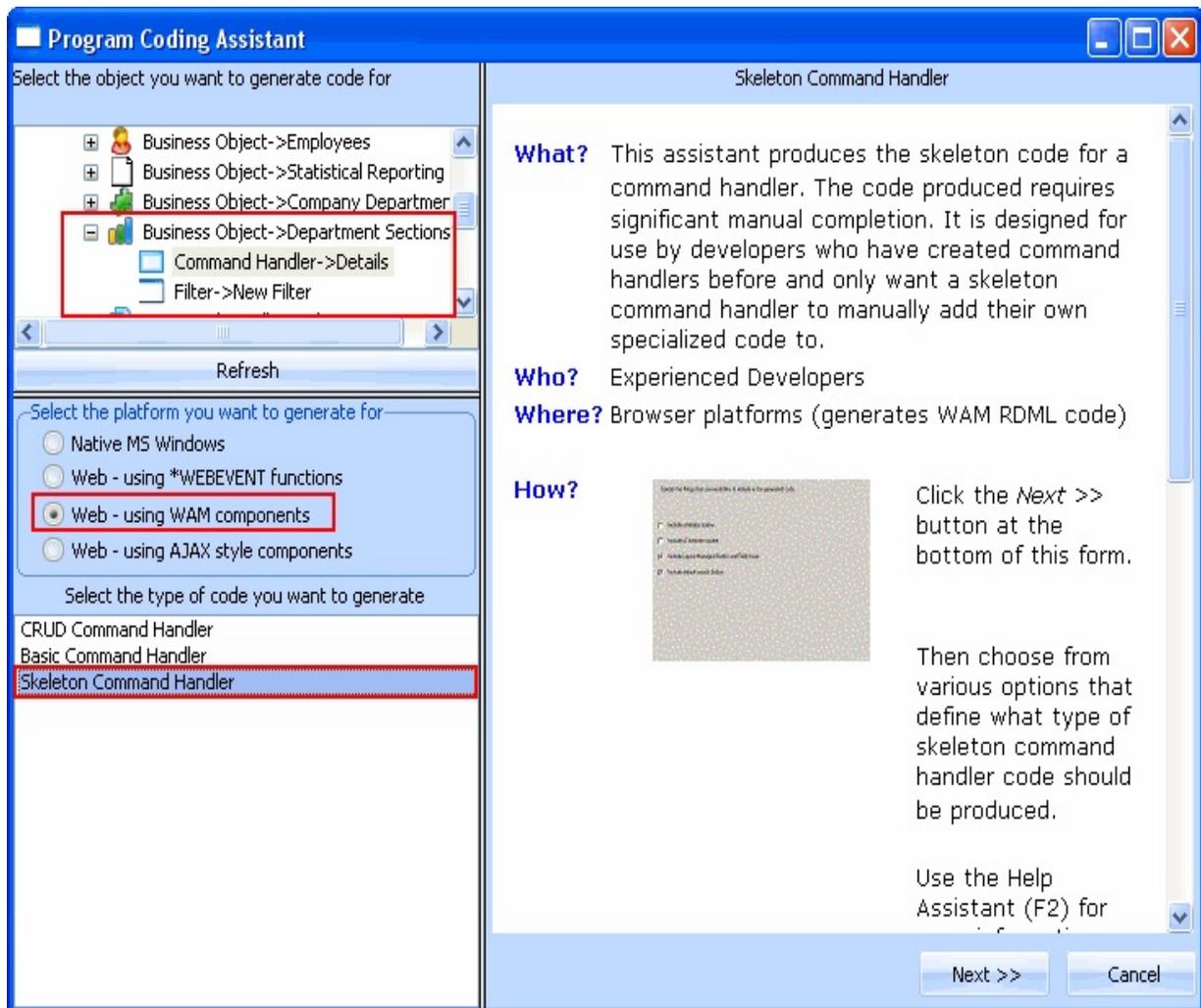


25. Close the application.

Step 2. Create a Details Command Handler for Department Sections

In this step you will create a Details command handler which will show the employees in the selected Section.

1. Start the Program Coding Assistant.
2. Select the Department Sections business object in the iii HR application.
3. Select the Command Handler -> Details.
4. Select Web – using WAM components as the platform.
5. Select Skeleton Command Handler:



6. Click Next.

7. Make sure the option Include Default Save Button and Logic is not selected.
8. Click Generate Code.
9. In the Generated Code page specify iiiCOM13 as the name of your command handler and Section Details as the description. (iii are your initials. If you are using an unlicensed or trial version of Visual LANSA, you must always use the three characters DEM to replace iii).
10. Click Create.
11. Wait until you see a message stating that the component has been created.
12. Switch to the Visual LANSA editor.
13. Locate the comment saying Map fields used in this form.
14. Add these statements underneath the comment:

```
Def_list Name(#WAM_LIST) type(*working) Fields(#EMPNO #SURNAME
#GIVENAME)
Web_Map For(*both) Fields(#WAM_LIST)
```

The WAM_LIST working list specifies the fields to be shown on the command handler. The Web_Map statement displays the fields.

Your code will now look like this:

```
* Map fields used in this form.
Def_list Name(#WAM_LIST) type(*working) Fields(#EMPNO #SURNAME #GIVENAME)
Web_Map For(*both) Fields(#WAM_LIST)
Override Field(#UB_PUSHB1) Default('Button 1')
Override Field(#UB_PUSHB2) Default('Button 2')
Web_Map For(*both) Fields((#UB_PUSHB1 *NOID)(#UB_PUSHB2 *NOID))
```

15. Add this code in the routine handling the #avFrameworkManager.uInitialize event to determine which section is selected:

```
Invoke #avListManager.GetCurrentInstance Found(#Ret_Code)
AKey1(#DEPARTMENT) AKey2(#SECTION)
```

Ignore the error message. Your code will now look like this:

```

* Initialize the command handler
EvtRoutine Handling(#avFrameworkManager.uInitialize) Options(*noclearmessages *noclearerrors)
  Invoke #avListManager.GetCurrentInstance Found(#Ret_Code) AKey1(#DEPARTMENT) AKey2(#SECTION)
  Name RET_CODE could not be found.
Endroutine

```

16. Add this code in the routine handling the #avFrameworkManager.uExecute event:

```

Define #Ret_Code reffld(#IO$STS)
Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')

Clr_list #WAM_LIST

Select fields(#WAM_LIST) from_file(PSLMST1) with_Key(#DEPARTMENT
#SECTION)
Add_Entry #WAM_LIST
EndSelect

```

If the return code from the avListManager.GetCurrentInstance method is OK, this code selects the employee fields from the PSLMST1 logical view and adds them to the command handler.

Your code will now look like this:

```

EvtRoutine Handling(#avFrameworkManager.uexecute) Options(*noclearmessages *noclearerrors)
  Define #Ret_Code reffld(#IO$STS)
  Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')

  Clr_List #WAM_LIST

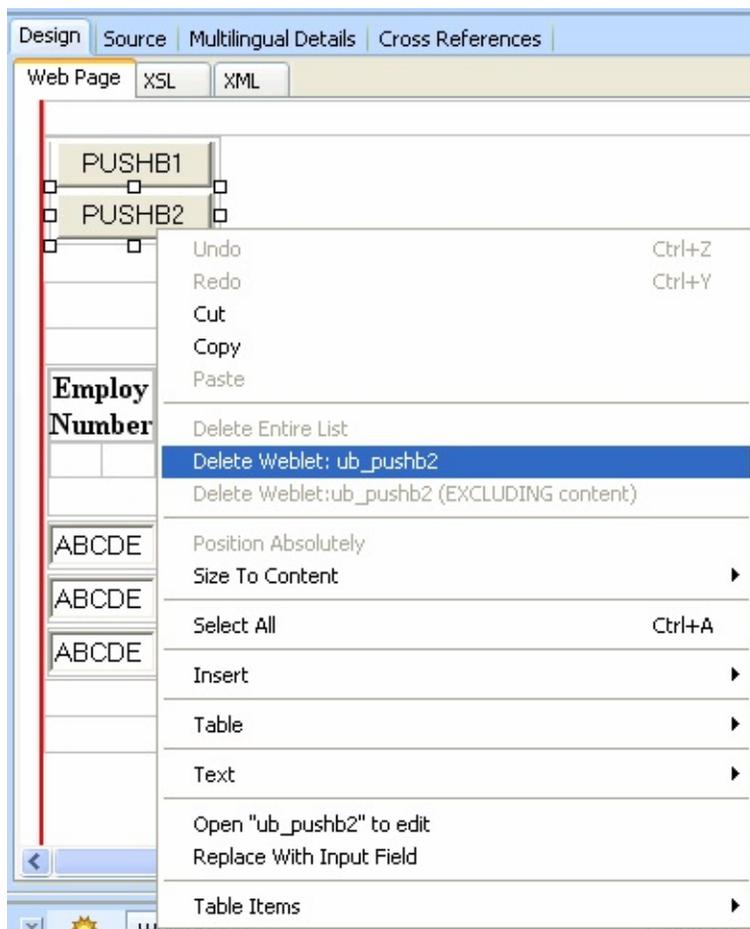
  EvtSelect fields(#WAM_LIST) from_file(PSLMST1) with_Key(#DEPARTMENT #SECTION)
    Add_Entry #WAM_LIST
  EndSelect
Endroutine

```

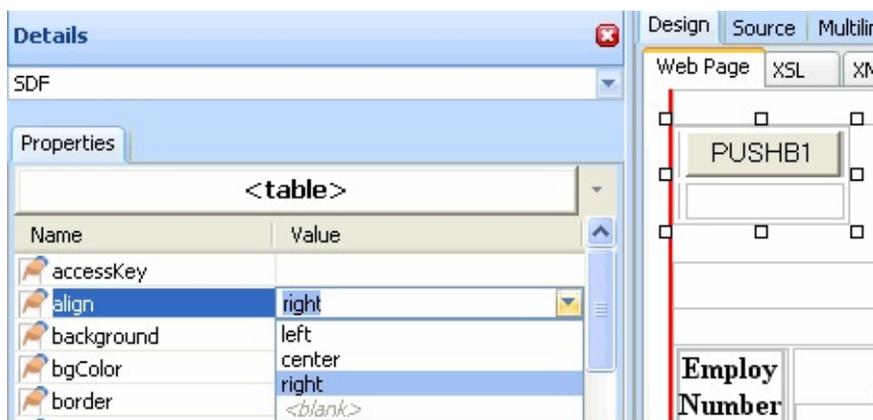
16. Compile the command handler.
17. Display the Design tab to see the command handler user interface.

You will only need one of the buttons created by the Program Coding Assistant in the command handler:

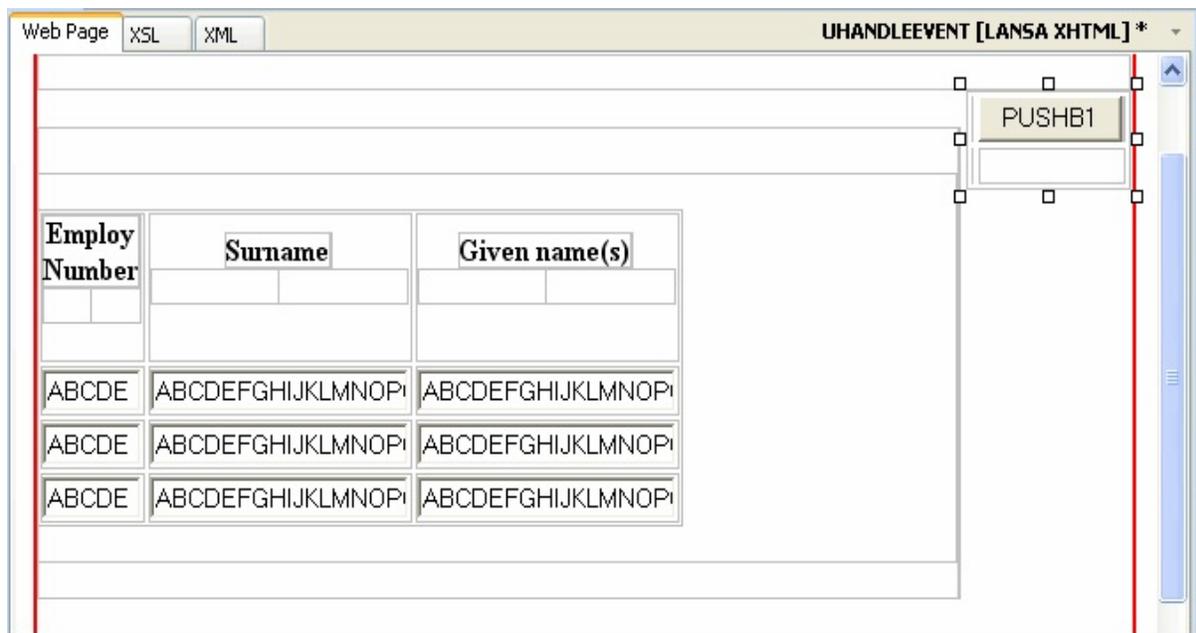
18. Select PUSHB2 and right-click, then select the option Delete Weblet:
ub_pushb2



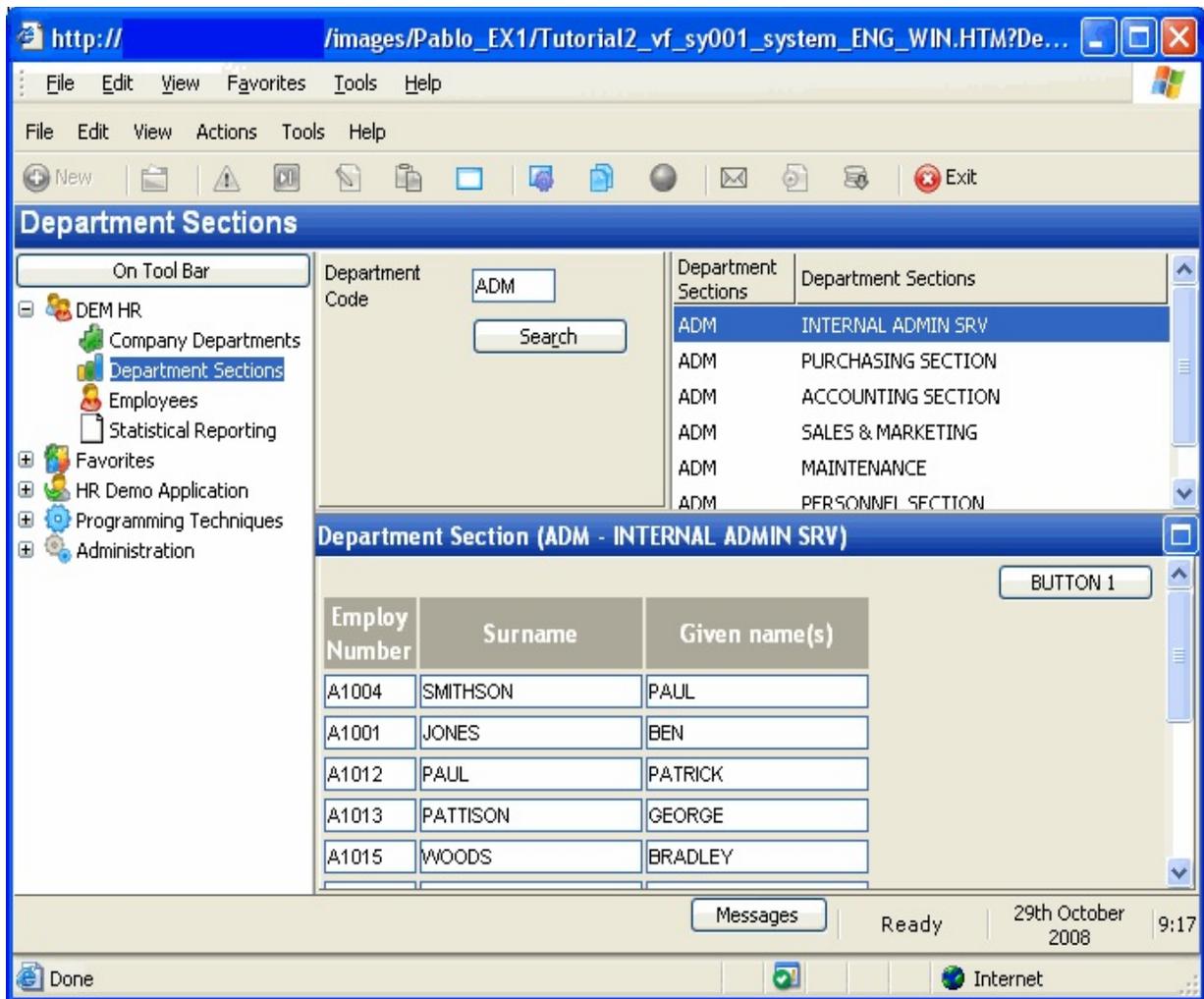
19. To align the remaining button on the right, select the table containing it, display the Details tab and set the Align property to right:



Your command handler now looks like this:



20. Save and compile the command handler.
21. Check the WAM and its associated layout weblet into the server with the Compile option selected.
22. In the Framework, snap the command handler in the Details command of the Department Sections business object.
23. Save the Framework and upload the details to the server.
24. Execute the Framework as a Web application.
25. Test your command handler.



26. Close the application.

Step 3. Add Logic to Switch from Sections to the Employees Business Object

In this step you will add logic to the Sections' Details command handler to display the details of a section's employees in the Details command handler of the Employees business object.

The switch to the Employees' Details command handler is executed in the button click event.

1. Display the Source tab of the Section Details command handler.
2. Locate the Override Field(#UB_PUSHB1) statement and change it to:

```
Override Field(#UB_PUSHB1) Default('Details')
```

3. In the #avFrameworkManager.uWAMEvent_1 eventroutine (handling the Details button click event) add this code to clear the instance list in the Employee Details command handler and populate it with the selected section's employees:

```
Invoke Method(#avListManager.BeginListUpdate)  
ForBusinessObject(EMPLOYEES)
```

```
Invoke Method(#avListManager.ClearList)
```

```
Select Fields(#EMPNO #GIVENAME #SURNAME) From_File(PSLMST1)  
With_key(#DEPARTMENT #SECTION) Generic(*yes) Nbr_Keys(*Compute)
```

```
Invoke #avListManager.AddtoList Visualid1(#EMPNO)  
Visualid2(#SURNAME) AColumn1(#givename)AKey1(#EMPNO)  
EndSelect
```

```
Invoke Method(#avListManager.EndListUpdate)
```

In WAM command handlers you can name the instance list you want to use, in this case the instance list for business object EMPLOYEES. (In Windows you use the avAddSwitchInstances event to work with another business object's instance list.)

4. Then add a statement to switch to the Details command handler of the Employees business object:

```
#avframeworkmanager.avSwitch To(BUSINESSOBJECT)
NAMED(EMPLOYEES) EXECUTE(DETAILS) Caller(#com_owner)
```

- The To parameter contains BUSINESSOBJECT to indicate the switch is to a business object (you can also switch to the Framework or an application).
- The NAMED parameter must contain your actual business object name.
- The EXECUTE parameter contains the name of the command to execute.

Your code will now look like this:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_1) Options(*noclearmessages *noclearerrors)
    * populate Employees instance list
    Invoke Method(#avListManager.BeginListUpdate) ForBusinessObject(EMPLOYEES)

    Invoke Method(#avListManager.ClearList)

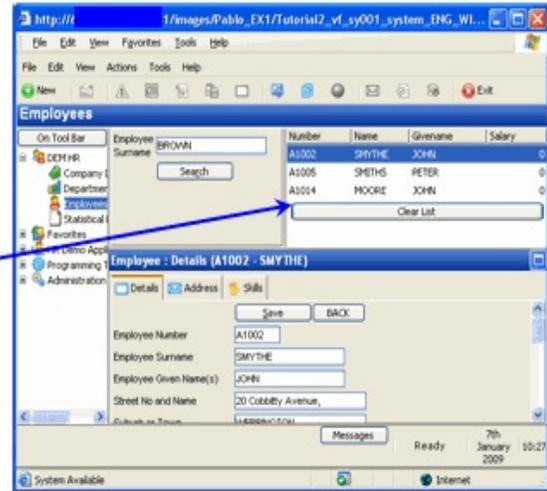
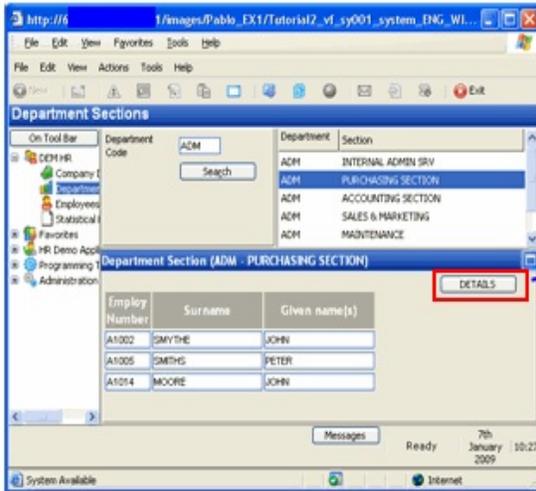
    Select Fields(#EMPNO #GIVENAME #SURNAME) From_File(PSLMST1) With_key(#DEPARTMENT #SECTION)
        Generic(*yes) Nbr_Keys(*Compute)
        Invoke #avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#SURNAME)
            AColumn1(#givename)AKey1(#EMPNO)
    EndSelect

    Invoke Method(#avListManager.EndListUpdate)

    * Switch to Employee Details
    #avframeworkmanager.avSwitch To(BUSINESSOBJECT) NAMED(EMPLOYEES) EXECUTE(DETAILS)
        Caller(#com_owner)

Endroutine
```

5. Compile the command handler and check it in.
6. Execute the Framework as a web application.
7. Test the switching: when you select a section and click on the Details button on the command handler, the Employees business object is displayed with the section's employees in the instance list. The details of the first employee in the list are shown.



8. Close the application.

Step 4. Record Switch History using the Virtual Clipboard

In this step you will record the switch history using the virtual clipboard so that the end-user will be able return to the object that initiated the switch.

To use the virtual clipboard most effectively you need to devise a standardized naming protocol for items that are posted onto it. In this exercise you will use this standard to store the switch history:

ID1 SWITCH_HISTORY

ID2 OBJECT_NAME/COMMAND_NAME

FromAValue <object or command name>

In effect you will be storing a switch history table on the clipboard. The first key or ID is the code SWITCH_HISTORY to indicate that all records with this ID are related to switching history.

The ID2 indicates whether you are switching to a business object or command. The actual business object name and command name are placed in the clipboard using the FromAValue parameter. You can use avobjecttype to get the current business object name and avcommandtype to get the current command name. You should not hard code these values.

1. Display the Source tab of the Sections' Details command handler.
2. In the #avFrameworkManager.uWAMEvent_1 event routine, before the avSwitch command, write this code to add the appropriate records to the switch history:

```
#avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY)
WithID2(OBJECT_NAME) FromAValue(#ThisHandler.avObjectType)

#avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY)
WithID2(COMMAND_NAME)
FromAValue(#ThisHandler.avCommandType)
```

Your code should now look like this:

```

EvtRoutine Handling(#avFrameworkManager.uWAMEvent_1) Options(*noclearmessages *noclearerrors)
  * Save to clipboard return list

  #avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY) WithID2(OBJECT_NAME)
    FromAValue(#ThisHandler.avObjectType)

  #avframeworkmanager.avsavevalue WithID1(SWITCH_HISTORY) WithID2(COMMAND_NAME)
    FromAValue(#ThisHandler.avCommandType)

  * populate Employees instance list
  Invoke Method(#avListManager.BeginListUpdate) ForBusinessObject(EMPLOYEES)

  Invoke Method(#avListManager.ClearList)

  Select Fields(#EMPNO #GIVENAME #SURNAME) From_File(PSLMST1) With_key(#DEPARTMENT #SECTION)
    Generic(*yes) Nbr_Keys(*Compute)

    Invoke #avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#SURNAME)
      AColumn1(#givename)AKey1(#EMPNO)
  EndSelect

  Invoke Method(#avListManager.EndListUpdate)

  * Switch to Employee Details
  #avframeworkmanager.avSwitch To(BUSINESSOBJECT) NAMED(EMPLOYEES) EXECUTE(DETAILS)
    Caller(#com_owner)
Endroutine

```

3. Compile the command handler and check it in.
4. Close the command handler.

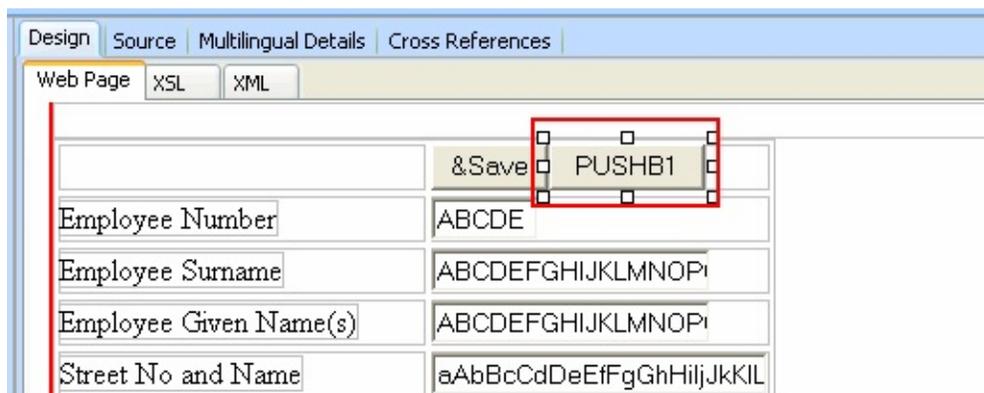
Step 5. Use the Switch History to Return to the Original Business Object

In this step you will use the switch history to allow the end-user to return to the Sections business object from where they initiated the switch.

1. Open the Employees' Details command handler iiiCOM06.
2. Display the Design tab.
3. In the Repository tab, locate Weblets, then the ub_pushb1 weblet.



4. Drag and drop the ub_pushb1 weblet next to the Save button in the command handler.



5. Display the Source tab.

6. Add this code before the UHandleEvent webroutine to set the caption of the button to Back:

```
Override Field(#UB_PUSHB1) Default('Back')  
Web_Map For(*both) Fields(#UB_PUSHB1)
```

7. In the UHandleEvent register the Back button Click event:

```
Invoke Method(#avFrameworkManager.avRegisterEvent)  
Named(UB_PUSHB1.CLICK) Signalaswamevent(2)
```

The UHandleEvent will now look like this:

```
Webroutine Name(UHandleEvent) ?  
  
* Register the event that will execute when clicking on the Save button in this filter as wam event number 1.  
Invoke Method(#avFrameworkManager.avRegisterEvent) Named(UB_SAVE.CLICK) Signalaswamevent(1)  
Invoke Method(#avFrameworkManager.avRegisterEvent) Named(UB_PUSHB1.CLICK) Signalaswamevent(2)  
* Standard WAM initialisation.  
  
Invoke Method(#avFrameworkManager.avInitializeWAM) Type(COMMAND) Invoker(#com_owner)  
Listmanager(#avListManager) Handlermanager(#ThisHandler) Fastpartmanager(#FastPart)  
Invoke Method(#avFrameworkManager.avHandleWAMEvent) Anchorblock(#vf_frame) Event(#vf_event)  
Designmode(#vf_framed) Skin(#VF_Frames) Metatag(#vf_elmeta) Tof(#vf_elxtof) N01(#vf_elxn01)  
N02(#vf_elxn02) N03(#vf_elxn03) N04(#vf_elxn04) N05(#vf_elxn05) N06(#vf_elxn06) N07(#vf_elxn07)  
N08(#vf_elxn08) N09(#vf_elxn09) N10(#vf_elxn10) A01(#vf_elxA01) A02(#vf_elxA02) A03(#vf_elxA03)  
A04(#vf_elxA04) A05(#vf_elxA05) A06(#vf_elxA06) A07(#vf_elxA07) A08(#vf_elxA08) A09(#vf_elxA09)  
A10(#vf_elxA10) Ssiname(#VF_FRAMEI)  
Endroutine
```

8. Next scroll to the end of the WAM source and add an event routine to handle the Back button Click event:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2)  
Options(*noclearmessages *noclearerrors)  
Endroutine
```

9. In this routine add this code so that when the users click on the button, they will be switched back to the business object from which they came:

```
define field(#ff_objnme) TYPE(*CHAR) LENGTH(32) DESC('Object  
Name')  
define field(#ff_cmdnme) TYPE(*CHAR) LENGTH(32) DESC('Command  
Name')
```

```
#avframeworkmanager.avrestorevalue WithID1(SWITCH_HISTORY)
WithID2(OBJECT_NAME) ToAValue(#ff_objnme)
#avframeworkmanager.avrestorevalue WithID1(SWITCH_HISTORY)
WithID2(COMMAND_NAME) ToAValue(#ff_cmdnme)

#avframeworkmanager.avSwitch To(BUSINESSOBJECT)
NAMED(#ff_objnme) EXECUTE(#ff_cmdnme) Caller(#com_owner)
```

- When you want to send the user back to the component from which the switch occurred, you need to look at the switch history on the virtual clipboard. Remember that you need to retrieve both the business object and the command to which you need to return. That requires retrieving two values from the virtual clipboard.
- The code first retrieves the OBJECT_NAME or business object value and then the COMMAND_NAME or command value.
- When you have these two values you can perform another switch to return to the previous component.

Your code should look like this:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2) Options(*noclearmessages *noclearerrors)
    define field(#ff_objnme) TYPE(*CHAR) LENGTH(32) DESC('Object Name')
    define field(#ff_cmdnme) TYPE(*CHAR) LENGTH(32) DESC('Command Name')

    #avframeworkmanager.avrestorevalue WithID1(SWITCH_HISTORY) WithID2(OBJECT_NAME) ToAValue(#ff_objnme)
    #avframeworkmanager.avrestorevalue WithID1(SWITCH_HISTORY) WithID2(COMMAND_NAME) ToAValue(#ff_cmdnme)

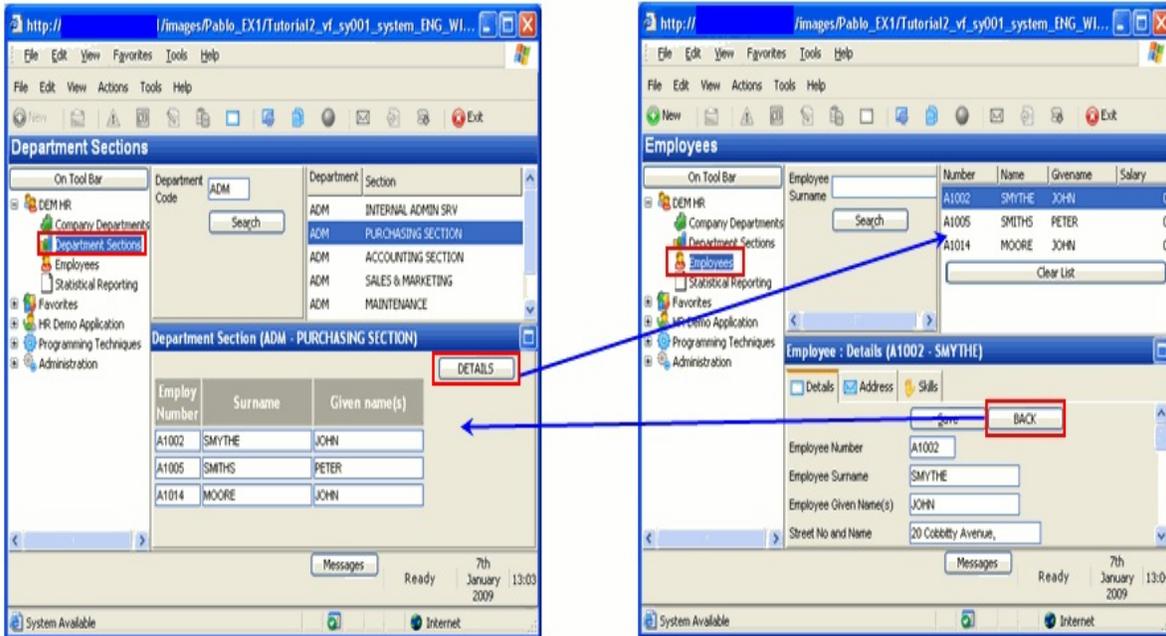
    #avframeworkmanager.avSwitch To(BUSINESSOBJECT) NAMED(#ff_objnme) EXECUTE(#ff_cmdnme) Caller(#com_owner)
ENDROUTINE
```

10. Compile the command handler and check it in.

You are now ready to test the switch history:

11. In the Framework select a section from the Department Sections business object.
12. Display the details of the section's employees by clicking on the Details button.
13. On the Details command handler of the Employees business object click on

the Back button to return to the Sections business object.



14. Close the application.

Summary

Important Observations

- The Framework switching service allows your filters and command handlers to switch control between different business objects and to execute commands at the Framework, application or business object level.
- The target business object must be able to be selected from the menu (the option Allow selection from the navigation pane in the target business object properties should be checked, and the user should be authorized to the business object), at the time the switch occurs. Switching mimics the actions that a user would perform.
- You can use the Virtual Clipboard for remembering and exchanging information.
- To use the virtual clipboard most effectively you need to devise a standardized naming protocol for items that are posted onto it.

Tips & Techniques

- The Advanced section of the Programming Techniques sample application has examples of switching and remembering values (virtual clipboard).

What I Should Know

- How to switch between business objects
- How to use the virtual clipboard to record switch history so that the end-users can switch back to object where the switch was initiated.

VLF013WAM - Signaling Events

Objectives

- To learn how to signal that an event has happened in your filter or command handler to other active filters or command handlers so that they can take appropriate action (see [Event Signaling Service](#)).
- To learn how to update an entry in the instance list (see [Filters and List Manager](#)).

In this tutorial you will add an event routine to the Employee Details command handler that signals employee details have changed.

You will then change the By Name filter to listen for this event and update the instance list.

Number	Name
A0070	BENTLEY VERONICA ANN
A1031	BLAIR JOHN
A0070	BLOGGS FRED JOHN ALAN
A3564	BROWN FREDDY

Clear List

A1031 - BLAIR JOHN

Skills

Save BACK

Employee Number: A1031

Employee Surname: BLAIR

Employee Given Name(s): JOHN

Street No and Name: 3 Woodbury Road

Suburb or Town: Winston Hills

State and Country: NSW Australia

Post / Zip Code: 2100

Messages Ready 9th January 2009 15:11

System Available Internet

To achieve this objective, you will complete the following steps:

Step 1. Change Employee Surname and Save the Changes

Step 2. Add the avSignalEvent to the Employee Details Command Handler

[Step 3. Add a Routine to Listen for the EMPLOYEE_CHANGED Event](#)

[Step 4. Test Signaling](#)

[Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

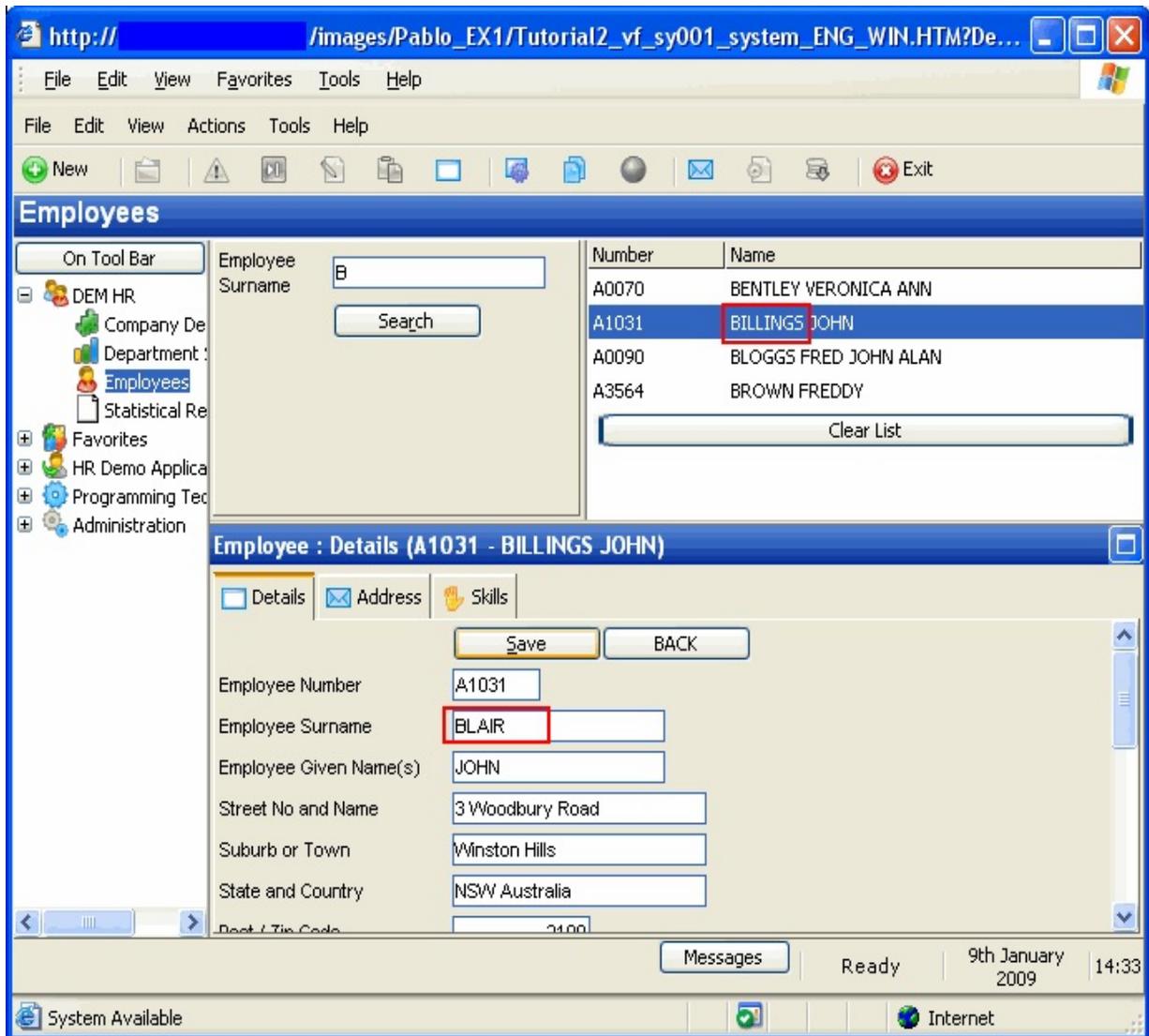
- Tutorials VLF000 – VLF007WAM and VLF009WAM

Step 1. Change Employee Surname and Save the Changes

In this step you will change the surname of an employee and save the changes. The instance list will not reflect the change because the filter does not know about the change event.

The By Location and By Date filters for the Employee business object are no longer required, so you can delete them before this step (if you have completed the Windows tutorials, you will already have removed them from your Framework design).

1. Display the Employees business object in the iii DEM application.
2. Use the By Name filter to add entries to the instance list.
3. Select one of the entries and change the employee surname in the Details command handler.
4. Click the Save button. Notice that the new surname is not reflected in the instance list entry:



In the following steps you will use the signal method in the Details command handler to notify the Employees filter that an employee has changed. You will then add code to update the instance list.

5. Close the application.

Step 2. Add the avSignalEvent to the Employee Details Command Handler

In this step you change the #avFrameworkManager.uWAMEvent_1 eventroutine in the Employee Details command handler to signal that employee details have changed.

1. Locate and open the Employee Details command handler iiiCOM06.
2. Display the Source tab.
3. Locate the #avFrameworkManager.uWAMEvent_1 eventroutine. Add the following code before the Endroutine statement:

```
Invoke #ThisHandler.avSignalEvent WithId(EMPLOYEE_CHANGED)  
SendAInfo1(#EMPNO) to(BUSINESSOBJECT)
```

Your code will look like this:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_1) Options(*noclearmessages *noclearerrors)  
    UPDATE FIELDS(#WAM_HEAD) IN_FILE(PSLMST) WITH_KEY(#EMPNO)  
    Invoke #ThisHandler.avSignalEvent WithId(EMPLOYEE_CHANGED) SendAInfo1(#EMPNO)  
        to(BUSINESSOBJECT)  
Endroutine
```

You use the avSignalEvent method when there is an event you would like other components within the Framework to be notified about:

- You place the event id to be signaled in the WithID parameter and any alphanumeric or numeric values you want to pass in the SendAInfoN or SendNInfoN parameters, where n is 1,...,5. In this example the event is EMPLOYEE_CHANGED and the employee number is the value to be passed.
- By default the value of the To parameter is FRAMEWORK which means any active component in the framework will receive this signal and will need to test to see if it pertains to them. If you know that the event only pertains to the business object in which this component resides, you should set the parameter To equal to BUSINESSOBJECT so that a very limited set of components are notified of this event. Using this technique will improve performance of your application.

4. Compile the command handler and check it into the server.
5. Close the command handler.

Step 3. Add a Routine to Listen for the EMPLOYEE_CHANGED Event

In this step you will change the Employees filter to listen for the EMPLOYEE_CHANGED event.

1. Locate and open the Employees filter iiiCOM04.
2. Display the Source tab.
3. Register the EMPLOYEE_CHANGED event in the UHandleEvent webroutine:

```
Invoke Method(#avFrameworkManager.avRegisterEvent)
Named(EMPLOYEE_CHANGED) Signalaswamevent(2)
```

The UHandleEvent will now look like this:

```
Webroutine Name(UHandleEvent)

* Register the event that will execute when clicking on the Search button in this filter as wam event number 1.
Invoke Method(#avFrameworkManager.avRegisterEvent) Named(UB_SEARCH_CLICK) Signalaswamevent(1)
Invoke Method(#avFrameworkManager.avRegisterEvent) Named(EMPLOYEE_CHANGED) Signalaswamevent(2)
* Standard WAM initialisation.

Invoke Method(#avFrameworkManager.avInitializeWAM) Type(FILTER) Invoker(#com_owner) Listmanager(#avListManager)
Filtermanager(#ThisFilter) Fastpartmanager(#FastPart)
Invoke Method(#avFrameworkManager.avHandleWAMEvent) Anchorblock(#vf_framew) Event(#vf_event)
Designmode(#vf_framed) Skin(#VF_Frames) Metatag(#vf_elmeta) Tof(#vf_elxtof) N01(#vf_elxn01) N02(#vf_elxn02)
N03(#vf_elxn03) N04(#vf_elxn04) N05(#vf_elxn05) N06(#vf_elxn06) N07(#vf_elxn07) N08(#vf_elxn08)
N09(#vf_elxn09) N10(#vf_elxn10) A01(#vf_elxA01) A02(#vf_elxA02) A03(#vf_elxA03) A04(#vf_elxA04)
A05(#vf_elxA05) A06(#vf_elxA06) A07(#vf_elxA07) A08(#vf_elxA08) A09(#vf_elxA09) A10(#vf_elxA10)
Ssiname(#VF_FRAMEI)
Endroutine
```

4. Scroll to the end of the WAM source and add an event routine to listen for the EMPLOYEE_CHANGED event signal:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2)
Withid(#eventid) WithAinfo1(#Ainfo1) Options(*noclearmessages
*noclearerrors)

Endroutine
```

5. In the event routine first add code to check the event id:

```
If '#EventId.Value = EMPLOYEE_CHANGED'
```

```
Endif
```

6. Then save the current key values from overwrites done by the select loop:

```
Inz_List #Save_Keys 1
```

7. Assign the value passed by the signaled event to the EMPNO field:

```
#EmpNo := #AInfo1
```

8. Start updating the instance list:

```
Invoke Method(#avListManager.BeginListUpdate)
```

9. Fetch the details of the employee that has been updated:

```
FETCH FIELDS(#SURNAME #GIVENAME #EMPNO)  
FROM_FILE(PSLMST) WITH_KEY(#EMPNO)
```

10. Update the entry in the instance list:

```
Use Builtin(BCONCAT) With_Args(#SURNAME #GIVENAME)  
To_Get(#FULLNAME)  
Invoke #avListManager.AddtoList Visualid1(#EMPNO)  
Visualid2(#FULLNAME) AKey1(#EMPNO)
```

11. Complete the instance list update:

```
Invoke Method(#avListManager.EndListUpdate)
```

12. Lastly restore the saved key values:

```
Get_Entry 1 #Save_Keys
```

Your finished code will look like this:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2) Withid(#eventid) WithAinfo1(#Ainfo1)
  Options(*noclearmessages *noclearerrors)
  If '#EventId.Value = EMPLOYEE_CHANGED'
    Inz_List #Save_Keys 1
    #EmpNo := #Ainfo1
    Invoke Method(#avListManager.BeginListUpdate)
    FETCH FIELDS(#SURNAME #GIVENAME #EMPNO) FROM_FILE(PSLMST) WITH_KEY(#EMPNO)
    Use Builtin(BCONCAT) With_Args( #SURNAME #GIVENAME) To_Get(#FULLNAME)
    Invoke #avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#FULLNAME) AKey1(#EMPNO)
    Invoke Method(#avListManager.EndListUpdate)
    Get_Entry 1 #Save_Keys
  endif
Endroutine
```

13. Compile the filter and check it in.

Step 4. Test Signaling

In this step you will test the signaling of the EMPLOYEE_CHANGED event.

1. Start the Framework as a web application.
2. Select the iii DEM application and the Employees business object.
3. Use the filter to populate the instance list.
4. Select an employee and change the surname.
5. Click Save. Notice that the filter now listens for the EMPLOYEE_CHANGED event and updates the list entry:

The screenshot shows a web browser window displaying a web application. The browser's address bar shows the URL: `http:// /images/Pablo_EX1/Tutorial2_vf_sy001_system_ENG_WIN.HTM?De...`. The application interface includes a menu bar (File, Edit, View, Favorites, Tools, Help) and a toolbar with icons for New, Open, Save, Print, and Exit. The main content area is titled "Employees" and features a search form on the left with a text input containing "B" and a "Search" button. To the right is a table listing employees:

Number	Name
A0070	BENTLEY VERONICA ANN
A1031	BLAIR JOHN
A0090	BLOGGS FRED JOHN ALAN
A3564	BROWN FREDDY

Below the table is a "Clear List" button. A secondary window titled "Employee : Details (A1031 - BLAIR JOHN)" is open, showing a form with tabs for "Details", "Address", and "Skills". The "Details" tab is active, displaying fields for Employee Number (A1031), Employee Surname (BLAIR), Employee Given Name(s) (JOHN), Street No and Name (3 Woodbury Road), Suburb or Town (Winston Hills), and State and Country (NSW Australia). A "Save" button and a "BACK" button are visible above the form fields. The bottom status bar shows "Messages", "Ready", the date "9th January 2009", and the time "15:17".

6. Close the application.

Summary

Important Observations

- The Framework manager provides a simple to use event signaling service that may be used in Windows or Web browser applications.
- To make event-processing work you need a filter or command handler that signals the event and other filters or command handlers that listen for the event. Additional information may be sent along with the event.
- To update an instance list entry you use the #avListManager.AddtoList method.

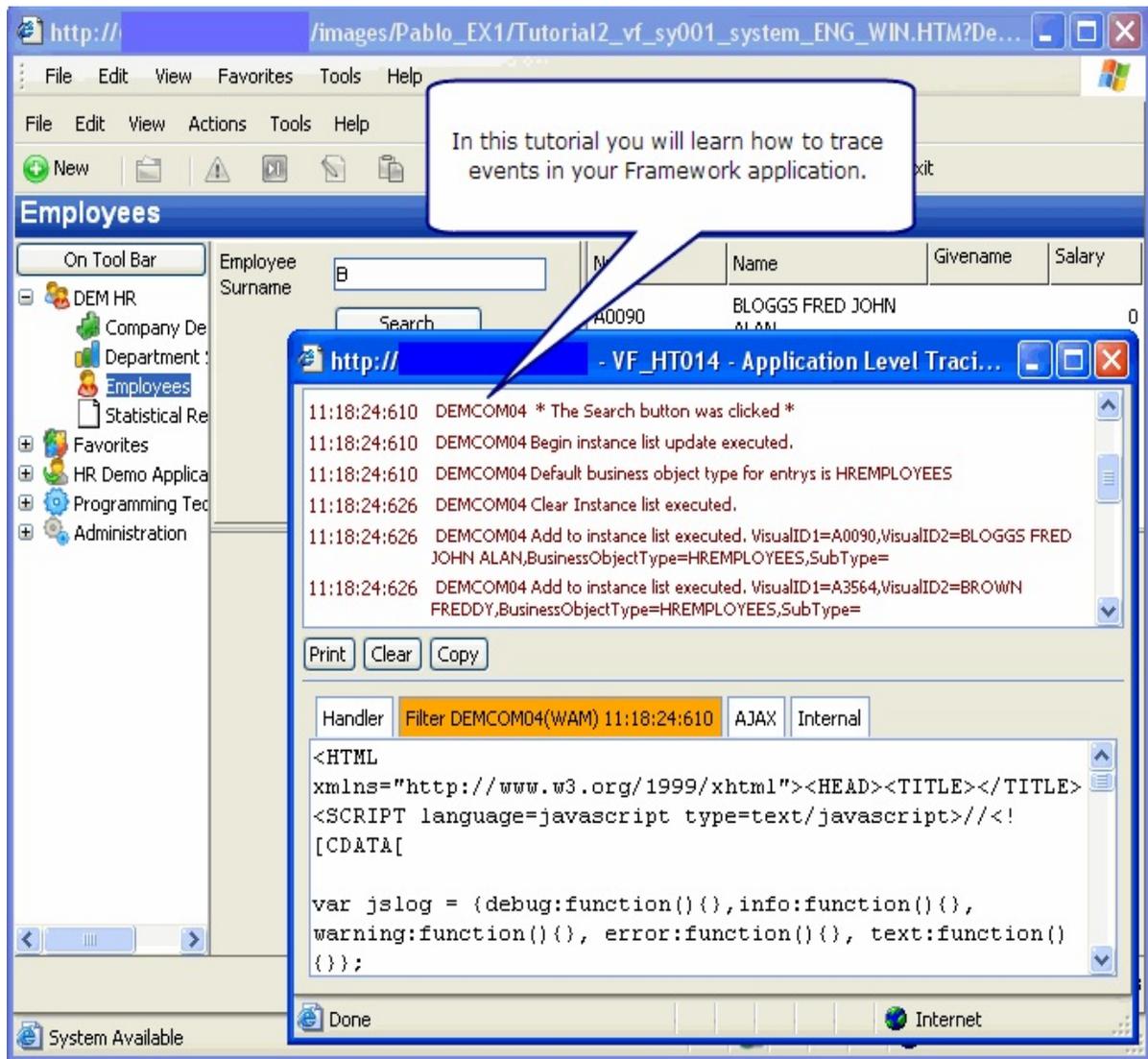
Tips & Techniques

What I Should Know

- How to signal an event
- How to listen for a signaled event
- How to update an entry in the instance list

VLF014WAM - Debugging/Tracing Objectives

To learn how to use the tracing service to help you locate problems in your filters or command handlers. (See [Basic Tracing Service](#).)



To achieve this objective, you will complete the following steps:

[Step 1. Add a Trace Statement to Indicate Enter Key Was Pressed](#)

[Step 2. Add More Trace Statements](#)

[Summary](#)

Before You Begin

In order to complete this tutorial, you must have completed the following:

· Tutorials VLF000 – VLF007WAM.

Step 1. Add a Trace Statement to Indicate Enter Key Was Pressed

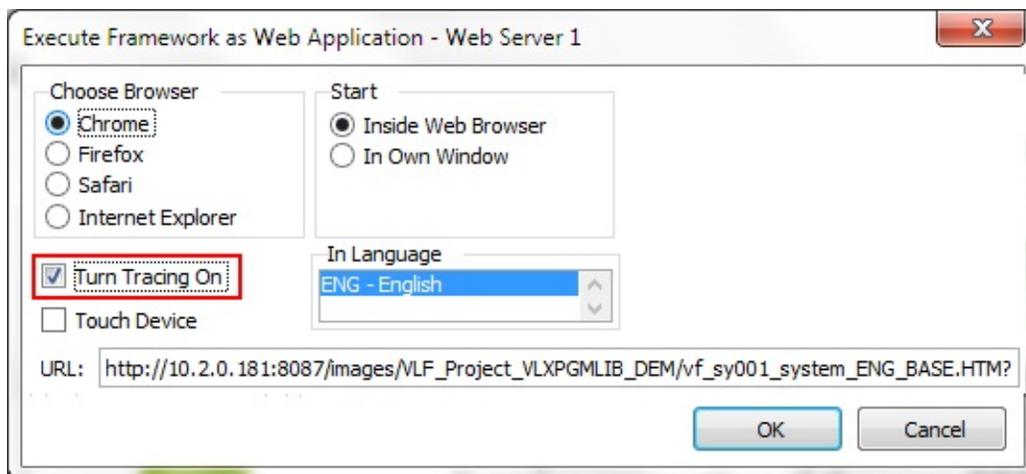
In this step, you will add a trace statement to show when the Search button was clicked in the Employees filter.

1. In the Visual LANSA editor open the Employees filter iiiCOM04.
2. Display the Source tab.
3. Locate the event routine handling Search button click event (#avFrameworkManager.uWAMEvent_1).
4. Add this tracing command in the beginning of the event handling routine:

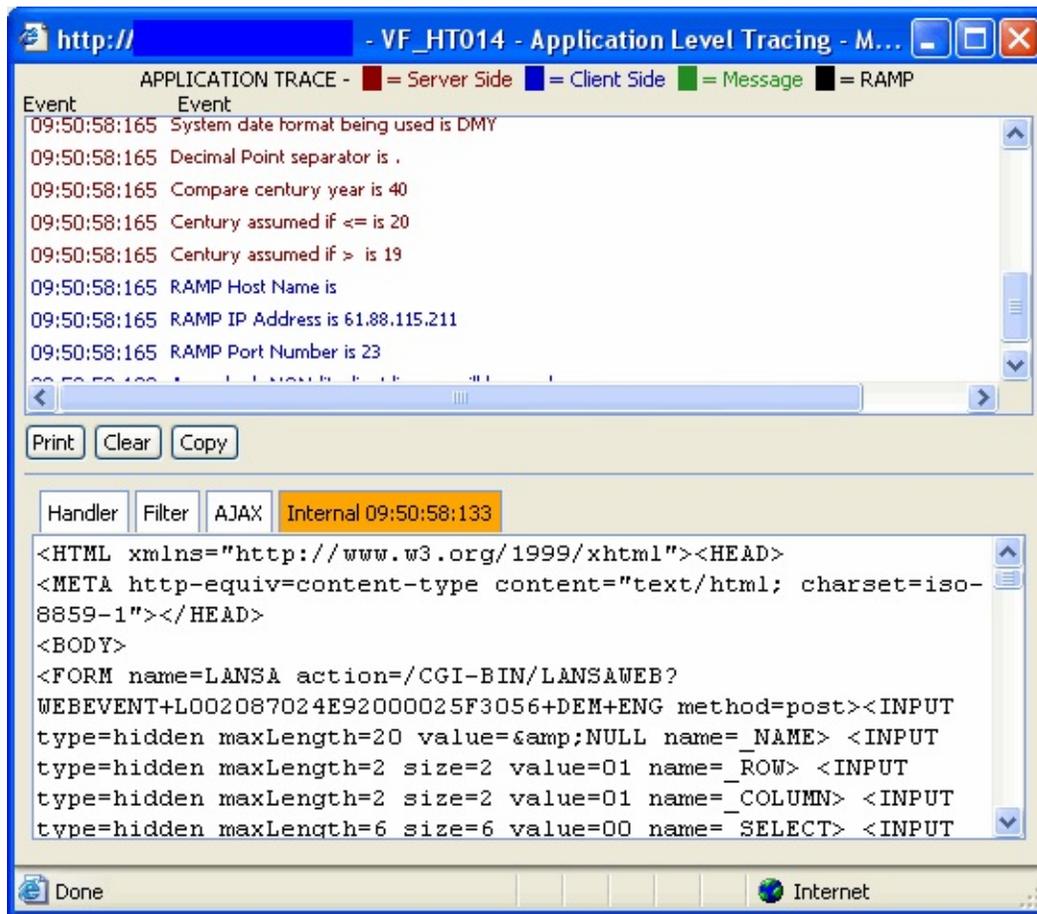
```
Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner)  
Event(' * The Search button was clicked *')
```

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_1) Options(*noclearmessages *noclearerrors)  
    Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner)  
        Event(' * The Search button was clicked *')  
    * If the search key(s) are valid  
    if *SearchOK
```

5. Compile the filter and check it into the server.
6. Start the Framework as a web application.
7. In the Execute Framework as a Web Application dialog select the Turn Tracing On option and click OK:



The Framework and the Trace Details windows are displayed:



The Tracing information displays detailed information about the execution of your Framework.

8. Expand the iii HR application and select the Employees business object. (Move the tracing window to the side if necessary).
9. In the Tracing window click on the Clear button to clear the trace information.
10. Type in a partial name in the Employee filter and click the Search button. The tracing window will now show that the Search button was clicked (scroll the trace details to locate your trace event).

http:// - VF_HT014 - Application Level Tracing - M...

APPLICATION TRACE - ■ = Server Side ■ = Client Side ■ = Message ■ = RAMP

Event	Event
09:56:38:535	Trace details cleared
09:56:40:117	Handling event UB_SEARCH.CLICK by executing WAM routine UHANDLEEVENT (in component DEMCOM04)
09:56:41:323	DEMCOM04 WAM started. Event is UB_SEARCH.CLICK
09:56:41:323	DEMCOM04 * The Search button was clicked *
09:56:41:323	DEMCOM04 Begin instance list update executed.
09:56:41:339	DEMCOM04 Default business object type for entries is HREMPLOYEES
09:56:41:339	DEMCOM04 Clear Instance list executed.
09:56:41:339	DEMCOM04 Clear Instance list executed.
09:56:41:339	DEMCOM04 Clear Instance list executed.

Print Clear Copy

Handler Filter DEMCOM04(WAM) 09:56:41:323 AJAX Internal

```
<HTML
xmlns="http://www.w3.org/1999/xhtml"><HEAD><TITLE></TITLE>
<SCRIPT language=javascript type=text/javascript>//

var jslog = {debug:function(){},info:function(){},
warning:function(){}, error:function(){}, text:function(){}};
var debug = function(){};

if (location.search.indexOf("enablejslog") != -1)
{</pre><p>Done Internet</p></div>
```

Step 2. Add More Trace Statements

In this step, you will add another trace statement to the filter. The statement will show when the EMPLOYEE_CHANGED event was signalled and the employee number passed.

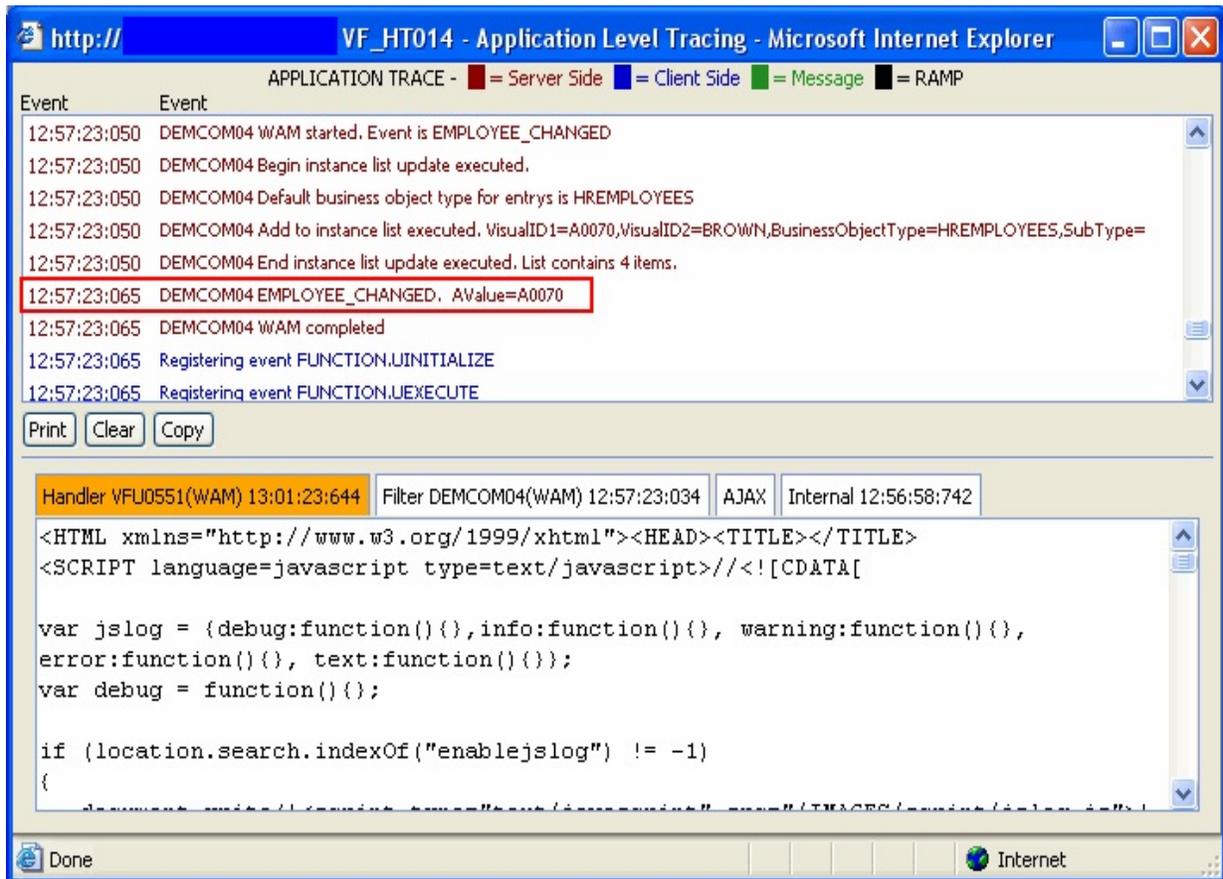
1. Locate the event routine handling the EMPLOYEE_CHANGED event (#avFrameworkManager.uWAMEvent_2) and add this statement to trace when the EMPLOYEE_CHANGED event is triggered and to show the employee number passed by the event:

```
#AVFRAMEWORKMANAGER.avRecordTraceAValue  
Component(#COM_OWNER) AValue(#EMPNO)  
Event(EMPLOYEE_CHANGED)
```

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2) Withid(#eventid)  
  WithAinfo1(#Ainfo1) Options(*noclearmessages *noclearerrors)  
  If '#EventId.Value = EMPLOYEE_CHANGED'  
    Inz_List #Save_Keys 1  
    #EmpNo := #Ainfo1  
    Invoke Method(#avListManager.BeginListUpdate)  
    FETCH FIELDS(#SURNAME #GIVENAME #EMPNO) FROM_FILE(PSLMST) WITH_KEY(#EMPNO)  
    Use Builtin(BCONCAT) With_Args( #SURNAME #GIVENAME) To_Get(#FULLNAME)  
    Invoke #avListManager.AddtoList Visualid1(#EMPNO) Visualid2(#FULLNAME)  
      AKey1(#EMPNO)  
    Invoke Method(#avListManager.EndListUpdate)  
    Get_Entry 1 #Save_Keys  
    #AVFRAMEWORKMANAGER.avRecordTraceAValue Component(#COM_OWNER) AValue(#EMPNO)  
    Event(EMPLOYEE_CHANGED)  
  endif  
Endroutine
```

2. Compile the filter and check it in.

3. In the Framework select employees using the filter and then select one employee to display the Employee Details command handler.
4. Change one of the employee details and click the Save button.
5. Notice the EMPLOYEE_CHANGED event and the employee number are shown in the trace:



Summary

Important Observations

- The Framework manager provides a basic tracing service to help you locate problems in your filters or command handlers.
- The tracing service can be used in conjunction with, or independently of, the normal LANSa application debugging and tracing facilities.

•

Tips & Techniques

- You can leave these method calls inside your code. The only time they have any effect is if tracing is turned on. Implementing tracing using this method is ideal as you don't have to remove the code at all if you do not wish to do so.
- The trace information can give you a lot of detailed information about what has happened which saves you having to run your application in debug mode.

What I Should Know

- How to trace Framework applications
- How to trace specific events in a filter or command handler

Tutorials for Deployment

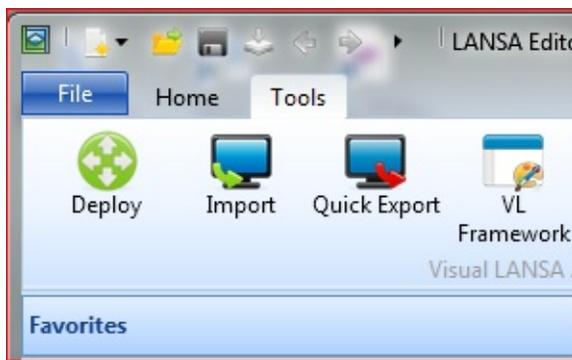
Applies to **Windows** only.

Includes:

[VLF008WIN - Deploying the Windows Framework](#)

When your application is complete, you use the Deployment Tool to create a package which will contain the software to be installed by your users.

The way you create your package varies depending on the environment it is to be used in. In this tutorial, you will learn how to deploy the Framework with the tutorial application for independent PCs.



The application is to run independently on every PC that it is installed on. Your users will be starting up the Framework as end-users.

The Deployment Tool is a feature of Visual LANS A and is not specific to the Visual LANS A Framework. For more details, refer to the LANS A Application Deployment Tool Guide.

Refer to [Visual LANS A Framework Deployment Check Lists](#) guide for detailed information on how to deploy your application.

VLF008WIN - Deploying the Windows Framework Objective

- Learn how to deploy your Windows Framework applications.
- To achieve this objective, you will complete the following steps:
 - [Step 1. Create the Package](#)
 - [Step 2. Specify the Startup Form and Database Type](#)
 - [Step 3. Disable all Prompting](#)
 - [Step 4. Add Framework Components](#)
 - [Step 5. Add Other Framework Objects](#)
 - [Step 6. Add Your Own Components](#)
 - [Step 7. Add the Data](#)
 - [Step 8. Add an Icon](#)
 - [Step 9. Check the Package](#)
 - [Step 10. Build the Package](#)
 - [Step 11. Ship the Package](#)
 - [Summary](#)

Before You Begin

Note that this tutorial is only an outline of steps and options to select to deploy the Framework and the tutorial application.

Your real application might contain objects other than your filters, command handlers, etc that must also be deployed. For example, if your application uses a Date handling BIF like CONVERTDATE you might have to add messages BIF0101 and BIF0102 from the LANSAs system message file DC@M01.

You may wish to review:

- The [Visual LANSAs Framework Deployment Check Lists](#) guide for detailed information on how to deploy your application..

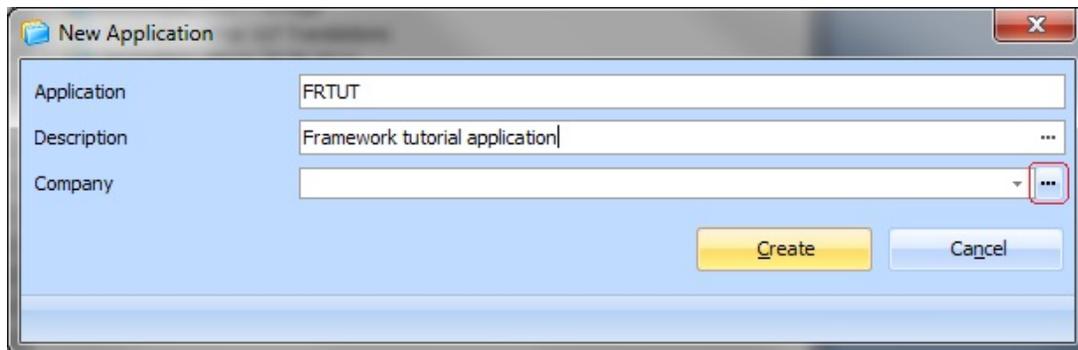
In order to complete this tutorial, you must have created an application ready to be deployed.

Step 1. Create the Package

To create a package for your software:

1. Start the Deployment Tool from the Tools tab on the ribbon.
2. Click on the New Application button.

The New Application dialog is displayed:

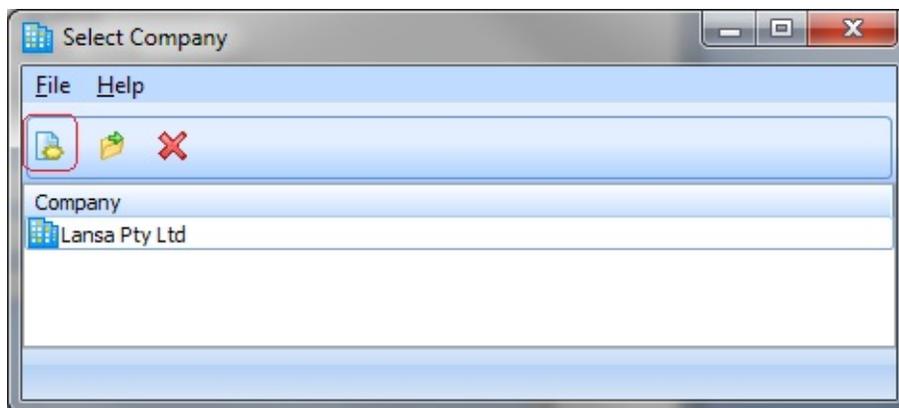


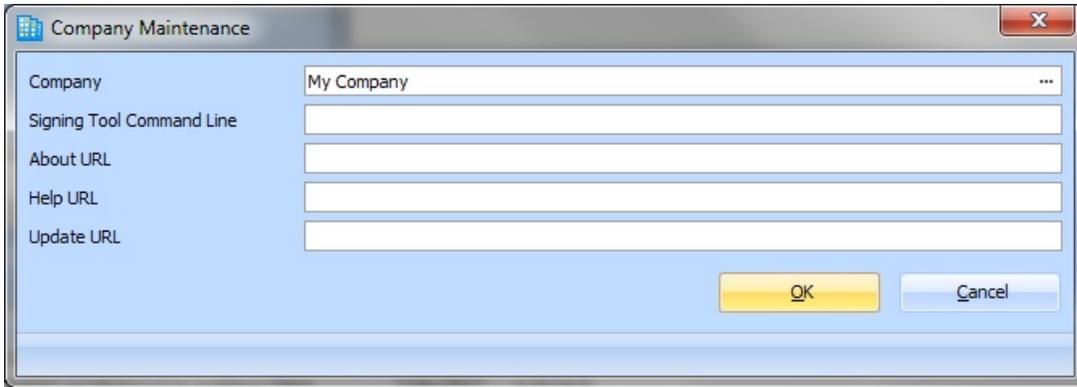
3. Create the new application with the name **iiiTUT** where **iii** are your initials.

Make its description **Framework tutorial application**.

4. If you have a company defined, select it.

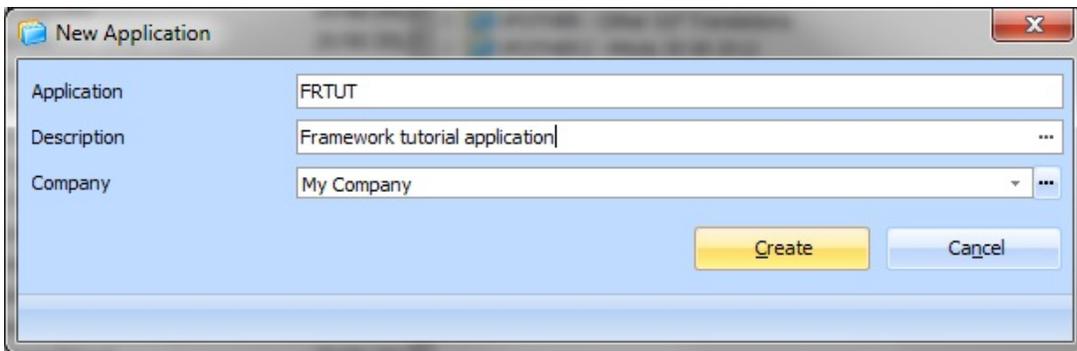
If you don't have a company, create a company using the company prompt and the new company button on the company dialog.



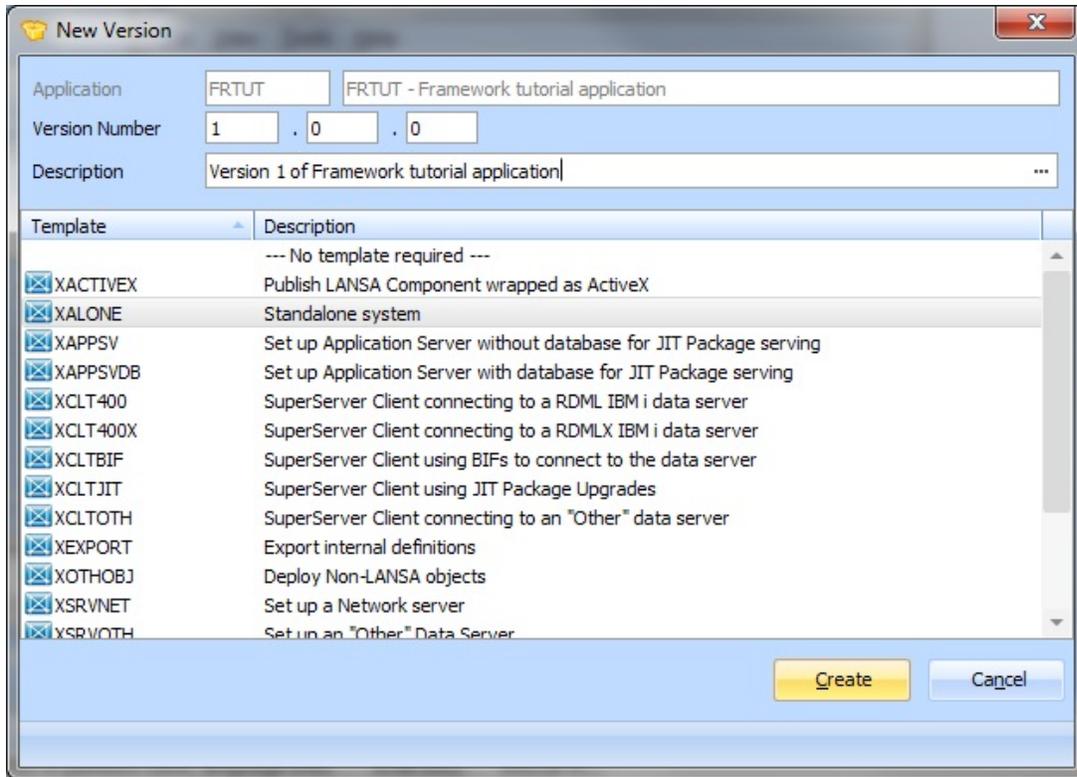


Click OK.

5. With the company selected, create the application.



The application is created and the New Package dialog is displayed:



4. Create a new package in the iiiTUT application using the package template called XALONE.

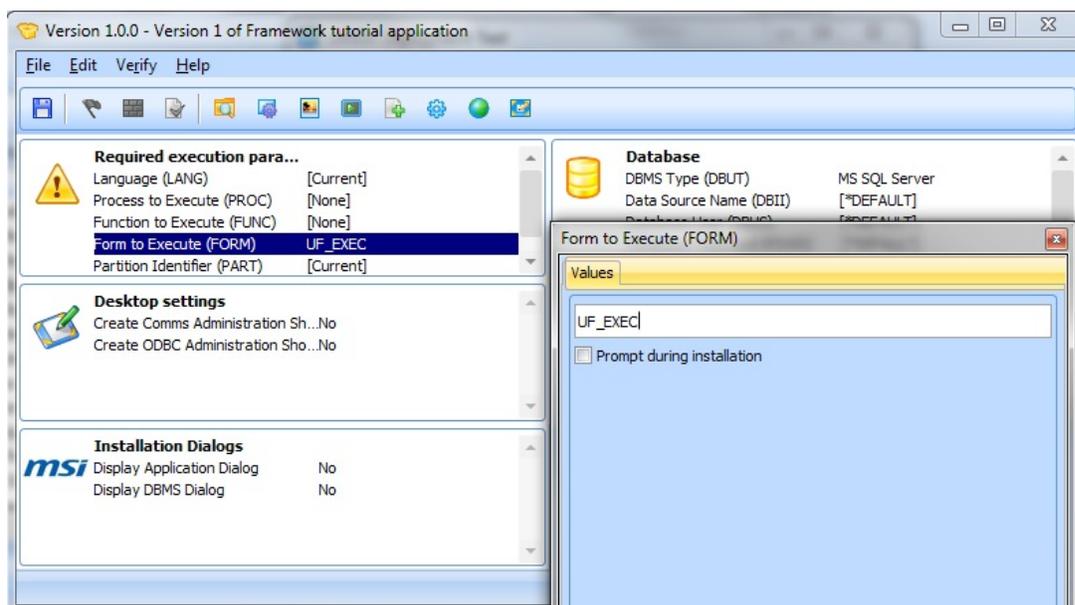
Make its description Version 1 of Framework tutorial application.

Click Create.

Step 2. Specify the Startup Form and Database Type

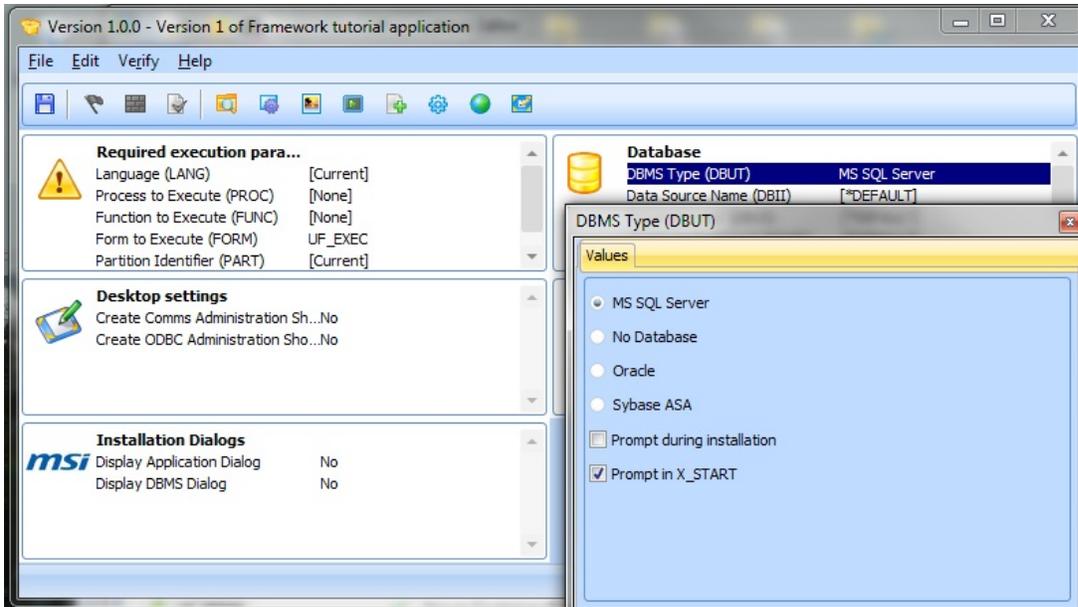
You will be deploying your application to end-users who will not require the design or administration facilities of the Framework. In this step, you will make the Framework run in user mode.

1. Click on the parameter Form to Execute (FORM=) in the Required execution parameters area.
2. Specify UF_EXEC in the window which is displayed.



3. Close the window.

By default the application will be deployed on a PC which has MS SQL Server database system installed. If the PC on which you plan to deploy the package has another type of database, specify it by clicking the DBMS Type parameter.



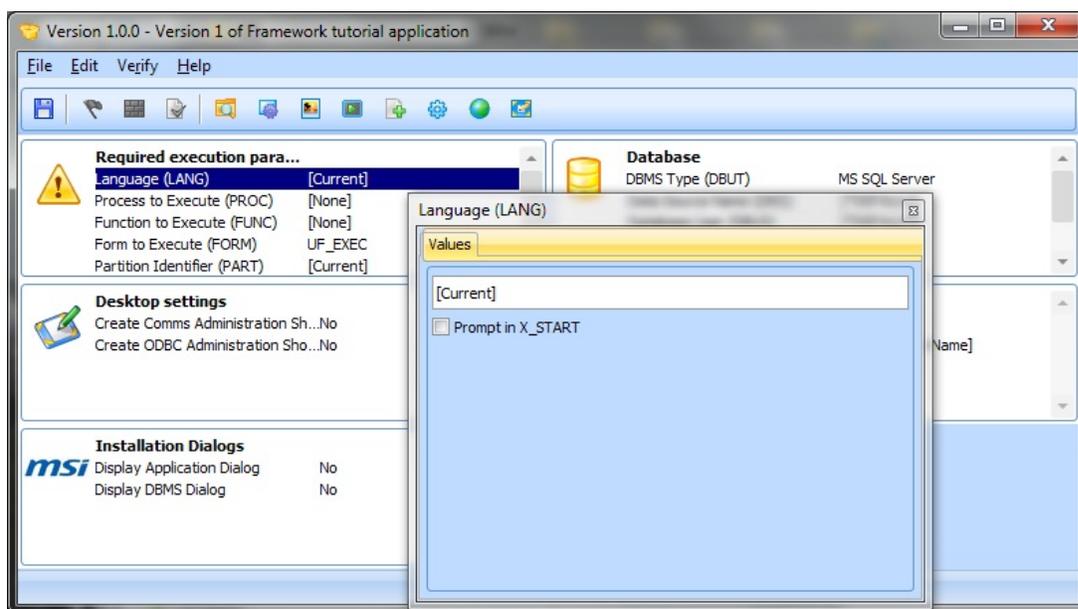
Step 3. Disable all Prompting

When the end-user executes your package, you do not want to prompt for any setup parameters. To disable the prompting, select the parameter and uncheck Prompt in X_START.

1. For example, disable the Prompting of the Current Language option.

Double-click the Language (LANG=) parameter.

Deselect the Prompt in X_START check box in the window which is displayed.



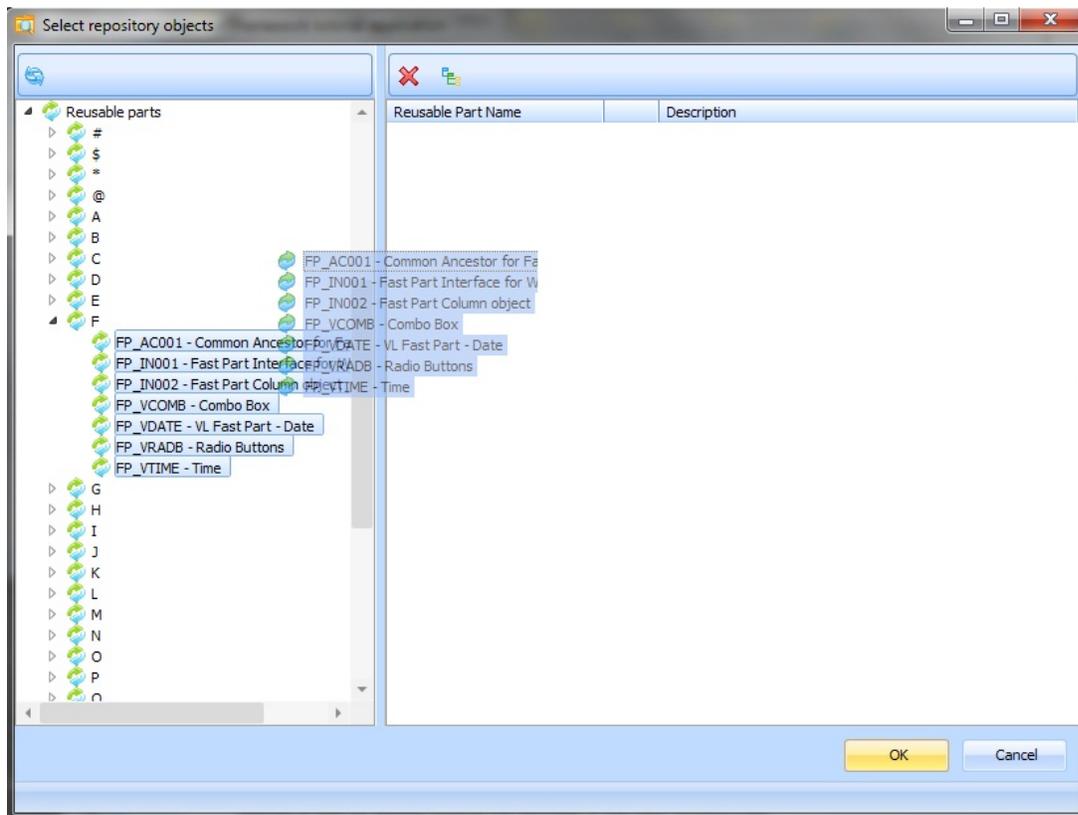
Close the window.

2. Check all the Required Execution Parameters and make sure the prompting is turned off for all of them!

Step 4. Add Framework Components

Next, you add the components that make up the Framework to the package.

1. Click on the Objects button on the toolbar.
2. Expand the Reusable parts node.



- a. Select all components starting with FP_* and drag them to the area on the right.
- b. Select all components starting with UF_* and drag them to the area on the right.
- c. Select all components starting with VF_* and drag them to the area on the right.

And if you want to be able to run the demonstration system:

- d. Select all components starting with DF_*, DM_*, DX_*, iiiCOM* and drag them to the area on the right.

3. Next select all the forms:

- a. Expand the forms.
- b. Select all components starting with FP_* and drag them to the area on the right.
- c. Select all components starting with UF_* and drag them to the area on the right.
- d. Select all components starting with VF_* and drag them to the area on the right.

And if you want to be able to run the demonstration system:

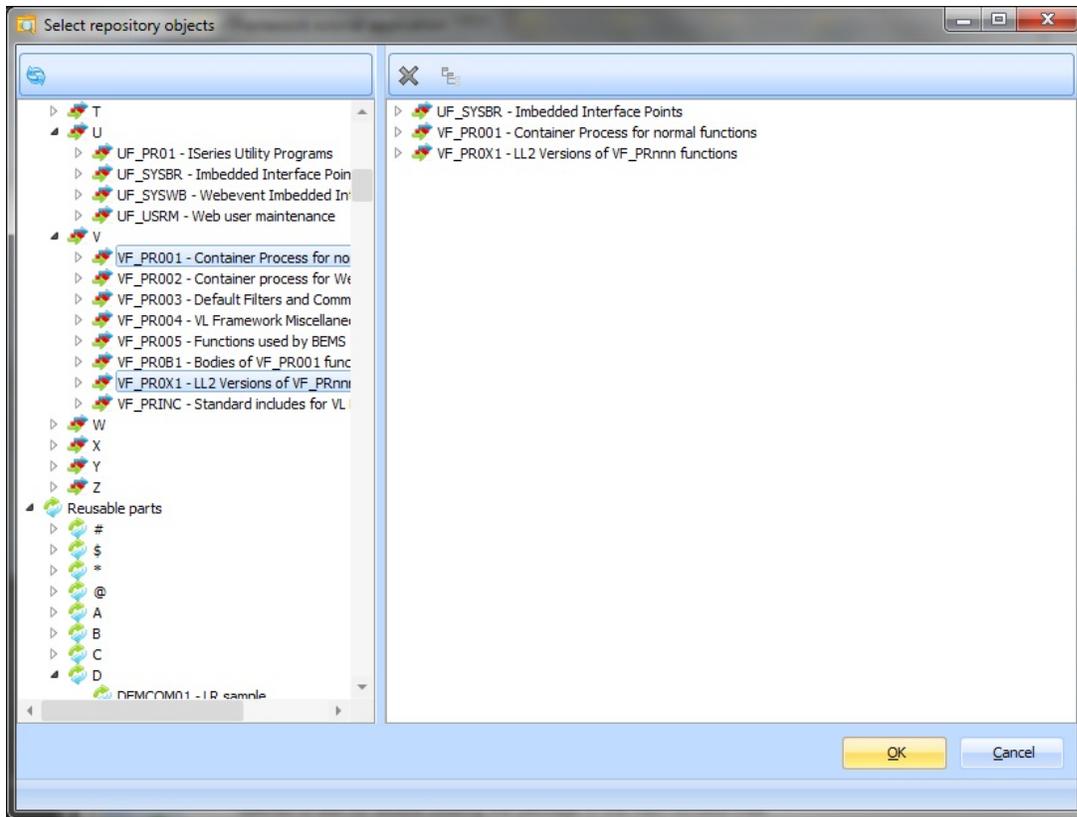
- e. Select all components starting with DF_*, DM_*, DX_*, iiiCOM* and drag them to the area on the right.

4. Next select processes and functions:

- a. Expand Processes/Functions.

Locate process UF_SYSBR and process VF_PR001 and VF_PROX1 and drag them to the area on the right.

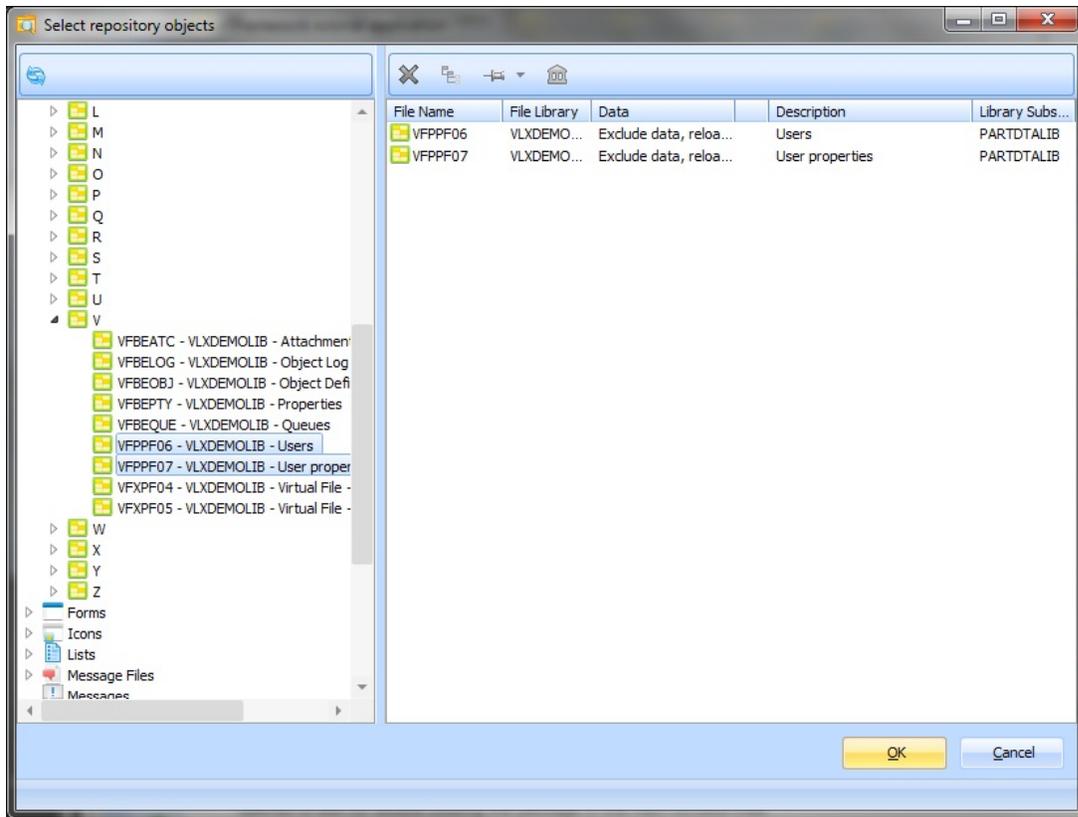
If you want to run the demonstration system, also add processes starting with: DF_*, DFX*, UF_*, VF_*, iiiPROC*.



5. If the end-user will be storing user information in DBMS tables VFPPF06/07 (See [Store Users in XML File and Store users in DBMS Tables VFPPF06/07](#)) you need to include them in the package:
 - a. Bring up the Other Objects tab.
 - b. Expand the Files node and then all files starting with V.

Select files VFPPF06 and VFPPF07 and drag them to the right-hand side of the window.

If you want to run the demonstration system, also add these files: DEPTAB, DXDOCS, FP*, PSL*, SECTAB and SKLTAB.

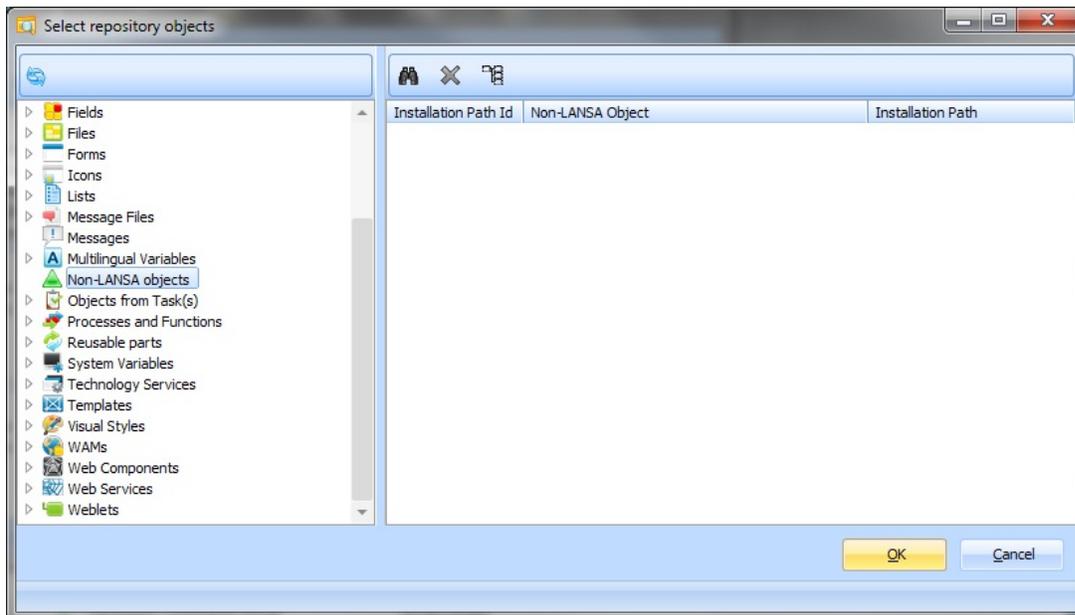


6. For a complete list of Framework objects see section [Framework Repository Objects](#) .

Step 5. Add Other Framework Objects

You also need to include some other Framework objects in the package.

1. Double-click Non-LANSAs Objects to browse for the objects..



2. Locate your XML files in the execute directory of your partition (for example C:\Program Files\LANSA\X_WIN95\X_LANSAs\x_DEM\execute).
3. Select VF_SY001_system.xml, vf_sy001_users.xml (if your application uses Framework Users and Authorities), vf_sy001_servers.xml, vf_sy001_tables.xml (if your application uses code tables).
4. Also select the splash screen for the HR system UF_im002.gif, and if you want to run the demonstration system, add df_demo*.htm, df_demo*.bmp, vf_ic*.gif.
5. If you are deploying the demonstration system, add vf_um823.htm, __DEFAULT__VF*, __PALLETTE__VF* and the RAD_PAD files for Statistical Reporting and Employees business objects.
6. Lastly select U_bif987.dll and u_bif985.dll from the LANSAs Execute directory.
7. Click the Add to Package button.

Step 6. Add Your Own Components

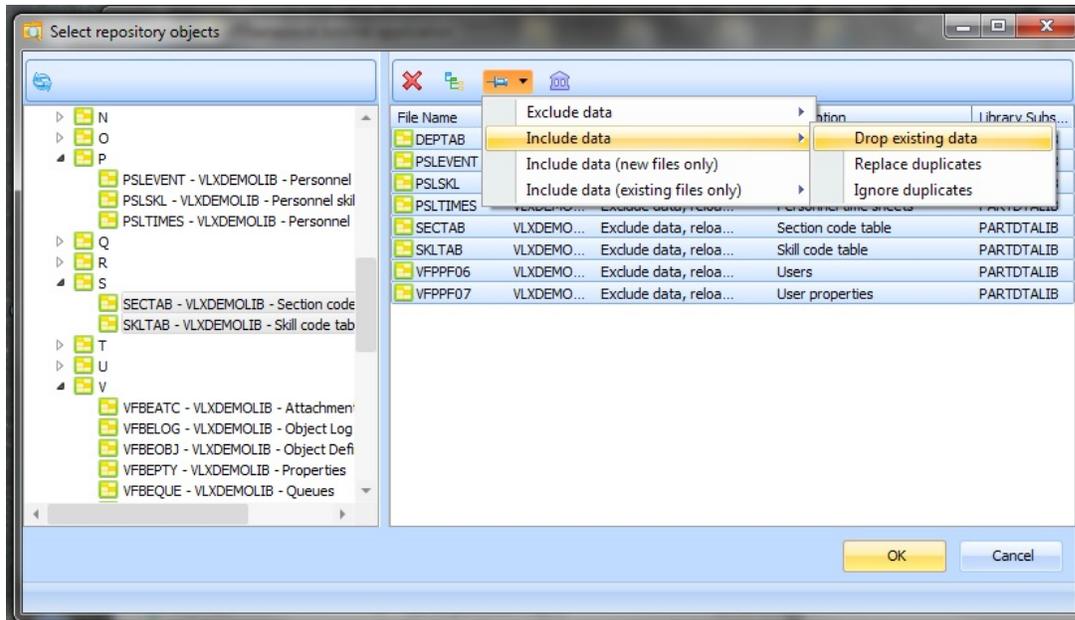
In this step, you will add all of your custom components (filters, command handlers etc.) used in the application.

1. Select Components in the Programs tab.
2. Choose the components which start with **iii** (**iii** being your initials) and drag them to the right-hand side of the window. These are your command handlers and filters.

Step 7. Add the Data

You also need to include the personnel files with data.

1. Expand Files in the Other Objects tab.
2. Locate the Personnel System files PSLMST, PLSKL, PSLEVENT, PSLTIMES, SECTAB, SKLTAB and DEPTAB and add them to the selected items.



3. Select the files and click on the Include File Data button.

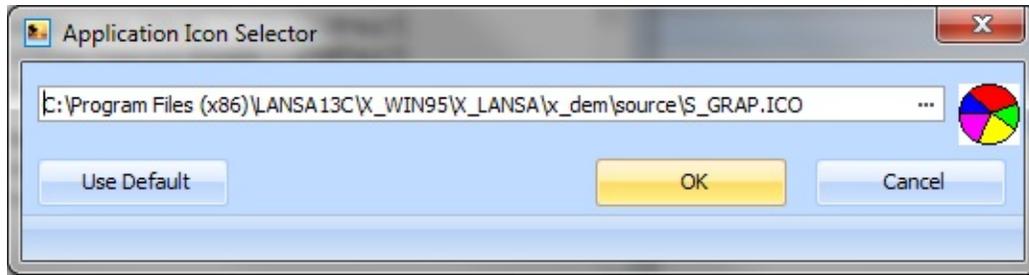
The Data column now indicates the files will be included with their data in the package.

4. Click OK.

Step 8. Add an Icon

You also need to specify an icon for the Framework program folder.

1. Click on the Icon button on the toolbar of the Package Control Panel.
2. Select an icon (for example, C:\Program Files\LANSA\X_WIN95\X_LANSA\x_DEM\source\S_grap.ico).

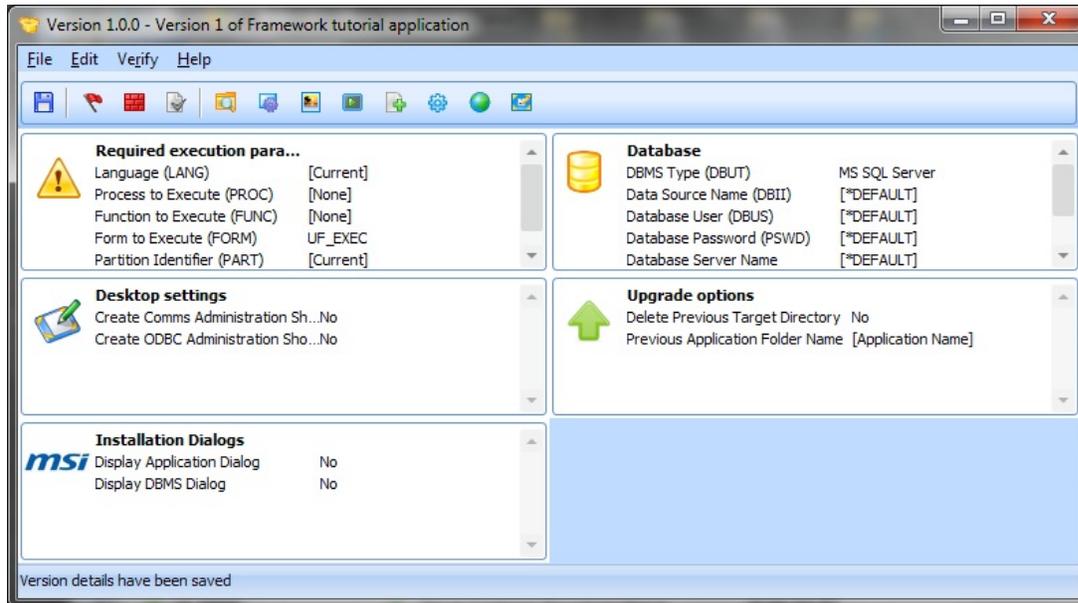


3. Click OK.

Step 9. Check the Package

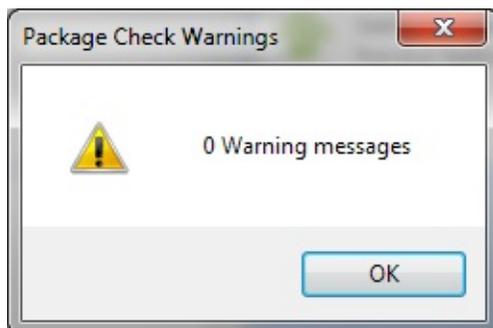
You are now ready to check the package and create it.

1. Save the package definition.



2. Select the Check Package option in the options menu.

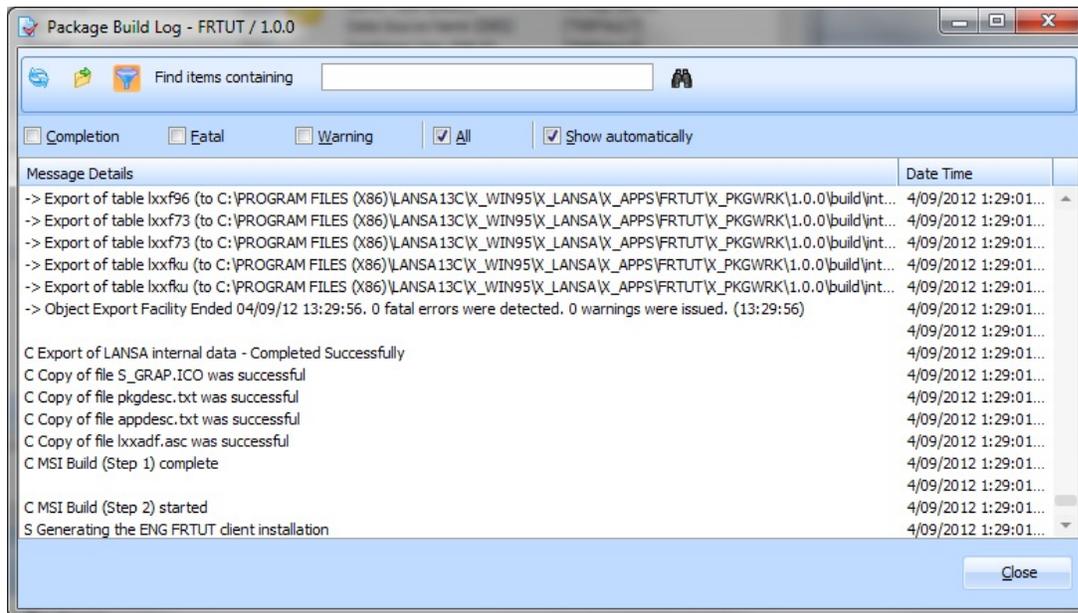
The message you receive should tell that no errors were encountered.



Step 10. Build the Package

You are now ready to build the package.

1. Click on the Build button.



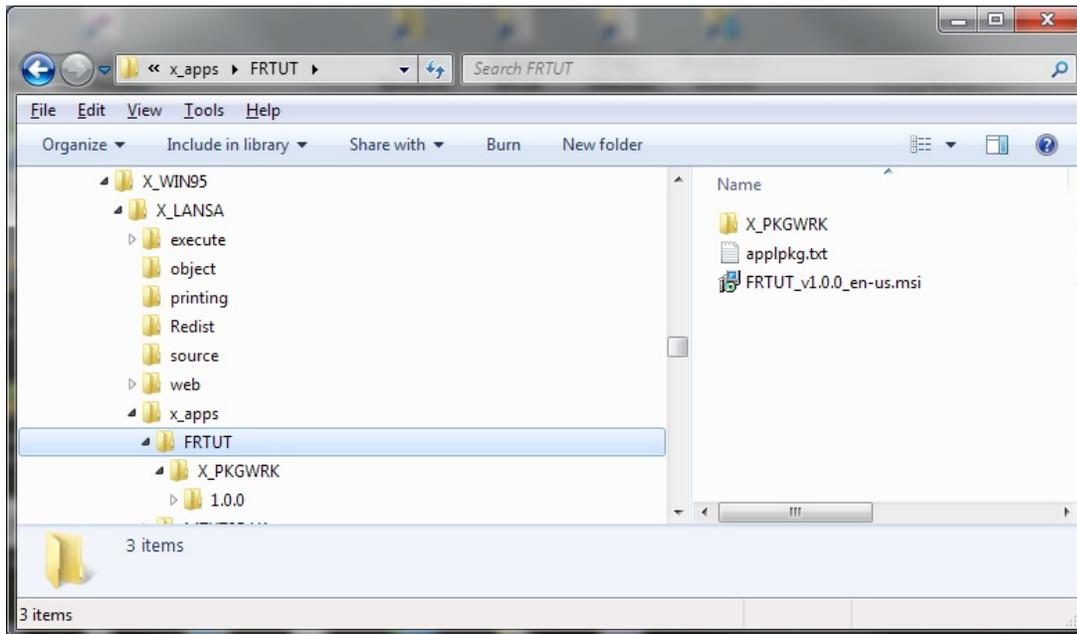
2. Wait for the package compilation to complete.

Step 11. Ship the Package

Your application package has been created in the X_APPS directory (for example. C:\Program Files\LANSA\X_WIN95\X_LANSA\X_APPS).

Note that the PCs on which this package is installed on must have the specified database system installed.

1. You will need to give your users the FRTUT_v1.0.0_en-us.msi file.
2. They will install the package by double clicking FRTUT_v1.0.0_en-us.msi.



The setup program automatically launches the Framework.

3. When the Framework is first launched executable files are copied and the Framework is displayed.
4. The users can now start using the application.

Summary

Important Observations

- When shipping your application created with the Visual LANSA Framework, you are shipping the actual Framework to the end-users (for end-user execution mode only).
- Remember to include your own custom objects (for filters and command handlers) when shipping your application.
- If shipping user information, you must remember to include the VFPPF06 and VFPPF07 files.
- This tutorial demonstrated the steps for shipping Visual LANSA Framework applications for Windows. The deployment of Web Framework applications were not part of this tutorial.
- The Deployment Tool is a feature of Visual LANSA and is not specific to the Visual LANSA Framework. For more details, refer to the LANSA Application Deployment Tool Guide.

What You Should Know

- How to deploy Framework applications on Windows.

Frequently Asked Questions

Framework presentation

How do I make the VLF use my corporate look?

How do I enroll my own visual styles?

How do I change a visual style at run time?

Can I define the toolbar by application?

How can I change the visual styles used in Web browser applications

I want to restore the default window layout for the Framework

How do I create an elaborate prototype?

Can I change the business object instance caption that appears in the area above my command handlers?

Can I stop VLF NET overriding CCS changes to background Colors of objects in WAMs?

Language

Text appears in English when I execute Framework as a web application

How can I translate text, for example the caption of the instance list Clear List button?

Pictures

How do I change Help About logos?

How do I change introduction logos?

How do I change the logo shown when the Framework starts executing?

How do I enrol my own bitmaps and icons?

Where does the Images Palette get the pictures from (when using the HTML formatted Images Palette)?

When I execute my Framework in a browser some of my icons and bitmaps don't display properly?

How are icons and bitmap names used when my Framework is executed in a browser?

How can I convert icons and bitmaps to GIF files?

Where do I need to put my GIFs so that my browser applications can use them?

Servers

How do I run my application in SuperServer mode?

Can my Web browser applications be used with System i multi-tier web server configurations?

Can my Web browser applications be used with Windows multi-tier web server configurations?

How do I pick up the servers that have been defined in the Framework using the Servers option of the Administration menu?

Changes not taking effect

I have created a new command but it does not appear in the menus or the toolbar?

I change Instance Command Presentation but the changes do not take effect?

I have just deleted an object but it is still visible?

I have just changed my Framework design but the change has not taken effect?

Why options I have disabled for Windows and/or the Web browser are displayed?

Versions and Upgrades

Are copies of my Framework design kept?

How can I change the list of Framework versions shown?

Can I use Visual LANSA Framework in Direct-X mode?

Settings

How are Framework settings remembered?

How are instance lists remembered?

Security

When I run the Framework security does not seem to work?

When I start the Framework as an administrator (for example UF_ADMIN), the Framework appears briefly and then disappears?

Do I have to (re)define my user profiles into the Framework?

Do I have to (re)define my user passwords into the Framework?

Are there techniques to minimize user profile (re)definition?

Web

What is a temporary directory and what is it used for?

How can I purge old information from my temporary directory?

Can I purge old information from my temporary directory in a batch job?

How can different users access different data libraries in Web applications?

Can I run web applications using secure sockets layer (SSL)?

How can I make my web applications launch into a new window?

How can I execute the VLF as a web application without using SSI?

How can I hide the address and status bars on Framework popup windows when using IE7?

The URL to start my deployed VLF web browser application is too complex for users to reliably type in to their browsers

What should I do when I get a mysterious WAM crash in my Framework RAMP application?

Why do I get a message saying 'Compression source file vf_multi_NAT.js not found'?

Why do I get a message saying 'Compression source file vf_multi_LLL.js not found'?

How can I tell if my WAM function is executing in a browser (VLF.WEB) or under .NET (VLF.NET)?

How to find out the L4Web images folder and path name in a VLF-WEB application?

How can users sign off from VLF-WEB sessions?

Can I customise the VLF WEB signon dialog?

Testing

I do my unit and suite testing in Design mode. Is this a good practice?

Other

How do I execute an initial Framework level command when starting up a Framework application?

I have just deleted an object and want to get it back again?

Why are some menus and menu options in brackets?

Can more than one person design the same Framework at the same time?

Can I change the options that the user searches for in the Quick Find box?

Can I control what happens when the user clicks on an option that they have located by using the Quick Find box?

Also See

Frequently Asked Questions (Users and Security)

[Frequently asked Questions about Code Tables](#)

[Frequently Asked Questions about Custom Properties](#)

Can I customise the VLF WEB signon dialog?

If you are comfortable with HTML and Javascript you can create your own customised signon window.

To implement please refer to file called UF_HT002_SAMPLE.js which is located in the partition execute folder. The instructions are included in the source.

When I include a jQuery Weblet in a VLF Wam it doesn't look right. What is wrong?

Maybe you haven't included a jQuery Theme to the WAM. The WAM guide has all the details about Theming WAMS but here are the basic steps:

1. Open the WAM and click on the Design tab.
2. With the design of the WAM displayed, select Manage External Resources from the Web menu.
3. Click the ADD button and select XWT01J Redmond - JQuery UI Widgets and XWT01L Redmond - LANSAs Theme Extensions. Make sure they appear in that order in the list. Click OK.
4. Save and re-run the WAM, it should display properly now.

Can I use Visual LANSAs Framework in Direct-X mode?

Yes you can. See [Using your Visual LANSAs Framework in Direct-X mode.](#)

Can I change the options that the user searches for in the Quick Find box?

Yes, see [Quick Find Override Feature](#).

Can I control what happens when the user clicks on an option that they have located by using the Quick Find box?

Yes, if you have provided the list of options to be searched. See [Quick Find Override Feature](#).

How to find out the L4Web images folder and path name in a VLF-WEB application?

```
#Folder := #Com_Owner.avImageFolder
```

```
#FolderPath := #Com_Owner.avImagePath
```

Can I change the business object instance caption that appears in the area above my command handlers?

Yes, you can.

In WAM applications do this:

```
Set #ThisHandler avHandlerCaption('My alternate caption')
```

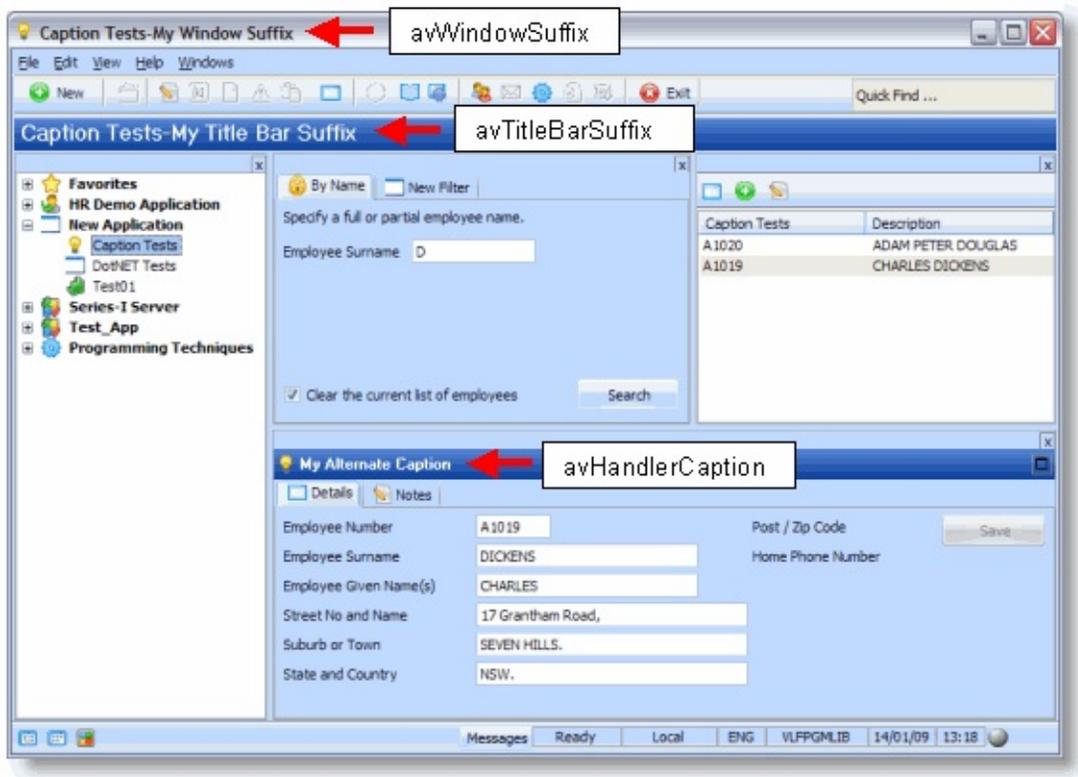
In WINDOWS applications do this:

```
Set #Com_Owner avHandlerCaption('My alternate caption')
```

In WINDOWS applications you can also add a suffix to the Window and title bar captions using this:

```
Set #Com_Owner avWindowSuffix('My Window Suffix')
```

```
Set #Com_Owner avTitleBarSuffix('My Title Bar Suffix')
```



You can't do this in RAMP because the command handler (VF_CH006) is a sealed program.

Can I stop VLF NET overriding CCS changes to background Colors of objects in WAMs?

Yes you can. See the LANSAs Support tip [Changing Background Colours of Objects in WAMs used in VLF .NET](#).

How can I tell if my WAM function is executing in a browser (VLF.WEB) or under .NET (VLF.NET)?

This is how you do it in a WAM:

```
If Cond(#ThisHandler.DotNET = TRUE)
Message Msgtxt('Execution platform on client is .NET')
Else
Message Msgtxt('Execution platform on client is web browser')
Endif
```

How do I execute an initial Framework level command when starting up a Framework application?

If you want to execute a specific command when a Framework starts, refer to [New UF_SYSTM IIP \(Imbedded Interface Point\) methods that you can override \(Windows\)](#) and the SwitchCommand parameter in [Web Application Start Options \(web\)](#).

Note that the concept of a default command at the Framework level has no meaning.

How do I make the VLF use my corporate look?

See [Personalize Your Framework](#).

How do I create an elaborate prototype?

Sometimes you may want something more than a rough mock up prototype in order to manage the end-users' initial impressions and expectations of your application. However, trying to produce elaborate filter or command handler layouts using mock ups can be a frustrating experience because a RAD-PAD is just simple HTML document with very limited editing capabilities.

In this situation it may be quicker and easier to create stub filters and command handlers and use the Visual LANSA IDE to produce much more sophisticated form layouts. These form layouts can then be completed at the end of the design cycle rather than discarded.

Here's an example of a simple stub filter, including the 'Emulate Search' button and logic:

```
*
=====
* Component   : XXXXXXXXXXXX
* Type        : Reusable Component
* Ancestor    : VF_AC007
* Created     : MM/DD/YYYY
* By          : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*
=====
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #VF_AC007) HEIGHT(132)
LAYOUTMANAGER(#ATLM_1) WIDTH(345)

DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_PANL) NAME(#BUTTON_PANEL)
DISPLAYPOSITION(1) HEIGHT(29) LAYOUTMANAGER(#FWLM_2)
LEFT(0) PARENT(#COM_OWNER) TABPOSITION(1) TABSTOP(False)
TOP(103) WIDTH(345)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_2)
ATTACHMENT(Bottom) MANAGE(#BUTTON_PANEL)
PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_PANL) NAME(#BODY_PANEL)
DISPLAYPOSITION(2) HEIGHT(103) LAYOUTMANAGER(#ATLM_2)
LEFT(0) PARENT(#COM_OWNER) TABPOSITION(2) TABSTOP(False)
TOP(0) WIDTH(345)
```

```

DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_3)
ATTACHMENT(Center) MANAGE(#BODY_PANEL) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_FWLM) NAME(#FWLM_1)
FLOWOPERATION(Center) FLOWOPERATIONHOR(Center)
FLOWOPERATIONVER(Center)
DEFINE_COM CLASS(#PRIM_FWLM) NAME(#FWLM_2)
FLOWOPERATION(Center) FLOWOPERATIONHOR(Decrease)
FLOWOPERATIONVER(Center) MARGINTOP(4) SPACING(4)
SPACINGITEMS(4)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#SEARCH)
CAPTION(*MTXTVF_EMUL_SEARCH) DISPLAYPOSITION(1)
LEFT(260) PARENT(#BUTTON_PANEL) TABPOSITION(1) TOP(4)
WIDTH(85)
DEFINE_COM CLASS(#PRIM_FWLI) NAME(#FWLI_4)
MANAGE(#SEARCH) PARENT(#FWLM_2)
Define_Com Class(#VF_UM822) Name(#VF_UM822) Reference(*Dynamic)
Define_Com Class(#VF_UM823) Name(#VF_UM823) Reference(*Dynamic)

DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_2)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) PARENT(#ATLM_2)

Evtroutine Handling(#SEARCH.Click) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
Define Field(#LIMIT) Reffld(#STD_NUM) Default(10)
Invoke Method(#uSystem.EmulateObjectSearch) Size(#Limit)
Avlistmanager(#avListManager) Instance(#puVF_FP503Owner)
Endroutine

End_Com

```

Here's a simple stub command handler you might use as a starting point:

```

*
=====
* Component   : XXXXXXXXXXXX
* Type        : Reusable Component
* Ancestor    : VF_AC010
* Created     : MM/DD/YYYY

```

* By : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
*

```
=====
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #VF_AC010) HEIGHT(193)
LAYOUTMANAGER(#ATLM_1) WIDTH(492)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_PANL) NAME(#BUTTON_PANEL)
DISPLAYPOSITION(1) HEIGHT(193) LAYOUTMANAGER(#FWLM_2)
LEFT(400) PARENT(#COM_OWNER) TABPOSITION(1) TABSTOP(False)
TOP(0) WIDTH(92)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_2)
ATTACHMENT(Right) MANAGE(#BUTTON_PANEL)
PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_PANL) NAME(#BODY_PANEL)
DISPLAYPOSITION(2) HEIGHT(193) LAYOUTMANAGER(#ATLM_2)
LEFT(0) PARENT(#COM_OWNER) TABPOSITION(2) TABSTOP(False)
TOP(0) WIDTH(400)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_3)
ATTACHMENT(Center) MANAGE(#BODY_PANEL) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_FWLM) NAME(#FWLM_1)
FLOWOPERATION(Center) FLOWOPERATIONHOR(Center)
FLOWOPERATIONVER(Center)
DEFINE_COM CLASS(#PRIM_FWLM) NAME(#FWLM_2)
FLOWOPERATIONHOR(Center) MARGINTOP(4) SPACING(4)
SPACINGITEMS(4)
Define_Com Class(#VF_UM822) Name(#VF_UM822) Reference(*Dynamic)
Define_Com Class(#VF_UM823) Name(#VF_UM823) Reference(*Dynamic)

DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_2)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) PARENT(#ATLM_2)

DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Button 1') DISPLAYPOSITION(1) LEFT(6)
PARENT(#BUTTON_PANEL) TABPOSITION(1) TOP(4)
DEFINE_COM CLASS(#PRIM_FWLI) NAME(#FWLI_1)
MANAGE(#PHBN_1) PARENT(#FWLM_2)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2)
```

```
CAPTION('Button 2') DISPLAYPOSITION(2) LEFT(6)
PARENT(#BUTTON_PANEL) TABPOSITION(2) TOP(33)
DEFINE_COM CLASS(#PRIM_FWLI) NAME(#FWLI_3)
MANAGE(#PHBN_2) PARENT(#FWLM_2)
```

```
EVTROUTINE HANDLING(#PHBN_1.Click)
OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES)
Use message_box_show (ok ok info *Component 'Button 1 has not been
implemented yet')
ENDROUTINE
```

```
EVTROUTINE HANDLING(#PHBN_2.Click)
OPTIONS(*NOCLEARERRORS *NOCLEARMESSAGES)
Use message_box_show (ok ok info *Component 'Button 1 has not been
implemented yet')
ENDROUTINE
```

```
End_Com
```

Do I have to (re)define my user profiles into the Framework?

Yes if you want to use the Framework's shipped object authority model. The Framework needs a way to associate user profiles with their allowable Framework activities.

Do I have to (re)define my user passwords into the Framework?

In i5/System i based applications the answer is generally no.

In System i/i5 based Framework applications, including RAMP, the validation of a password for a user profile can be performed by IBM i at the point the user accesses the i5/ System i system. For the Framework in general this is typically happens when the user starts a super-server session. For RAMP this is typically done when the user signs on to their 5250 session. In web browser based applications either web server user authentication or a user exit (IIP) can be used to validate the password via the operating system, rather than by the Framework itself.

In Windows applications the answer is generally yes, because user profile management in Windows products is generally much more fragmented.

Are there techniques to minimize user profile (re)definition?

Yes. These include the use of group users and role based users.

Using group users will not impact how many user profiles need to be defined, but it significantly impacts their ongoing maintenance. It is easier to maintain a single user group than a set of individual users. Note that the Framework allows a user to be a member of multiple groups, which is different to IBM i. You can use Framework user groups without having to use IBM i group profiles.

A role based user is where the real user is mapped to a much smaller set of Framework role based users.

For example real users Fred, Mary and Bill might all map to the single role based Framework user profile ACC_CLERK.

How can I execute the VLF as a web application without using SSI?

Use these scenarios to decide the one that best applies to you.

- I am in the process of configuring the Visual LANSA Framework to run as a Web application.

When running the Administrator's Console as part of the normal configuration steps, uncheck the Use Server Side Include option. When choosing this option you use client side transformations.

You must refer to [Enabling WAM Client Side XML Transforms in the >Web Application Start Options](#).

- I have been successfully executing the Visual LANSA Framework as a Web application. I would like to stop using SSI.

Execute the Administrator's Console and uncheck the Use Server Side Include option. Save and verify the values. When choosing this option you must use client side transformations. **You must refer to** [Enabling WAM Client Side XML Transforms in the >Web Application Start Options](#).

Can I run web applications using secure sockets layer (SSL)?

Yes, Framework web applications will run with Secure Sockets Layer (SSL). First see your HTML vendor's documentation regarding setting up SSL on your web server then see the LANSA for the Web documentation on using the Web Administrator to add another port for SSL (usually 443) with identical details to the main port and register a default user for the SSL port. With SSL running correctly on your web server simply change your application URLs to start with 'https://' instead of 'http://'.

When testing your SSL setup try accessing a simple HTML page first and then a LANSA for the Web function (eg. VF_PR004/VFU0401) before trying to run a Framework application under SSL.

How can I make my web applications launch into a new window?

If you launch Framework web applications from shortcuts or from the Execute as a Web application... menu option and your application loads into a existing IE window, it means you have the Reuse windows for launching shortcuts internet option checked.

To change this option open IE and from the Tools menu select Internet Options. Select the Advanced tab and from the Browsing group of options uncheck Reuse windows for launching shortcuts. Now when you launch a Visual LANSAs Framework web application it will start in a new window.

How can different users access different data libraries in Web applications?

All command handlers and filters have an initialisation call:

```
Include Process(*DIRECT) Function(VFSTART) [at the start of the function]
```

together with a close logic, initiated by a call to VFEND:

```
Include Process(*DIRECT) Function(VFEND) [at the end of the function]
```

Similarly we recommend you that you implement your own initialise and close calls. For example:

```
Include Process(*DIRECT) Function(MY_START) [at the start of the function]
Include Process(*DIRECT) Function(MYEND) [at the end of the function]
```

Ideally MY_START would be placed after VFSTART and MYEND would be placed before VFEND.

Inside your MY_START function, you can retrieve the value of the current logged on user by using the VF_GET feature with the loggedonuser keyword:

```
Use Builtin(VF_GET) With_Args('loggedonuser') To_Get(#STD_USER)
```

Then you need to call another piece of logic (for example a CL program) to set the library list for the user logged on.

When I start the Framework as an administrator (for example UF_ADMIN), the Framework appears briefly and then disappears?

To use UF_ADMIN (or equivalent) the user must also be flagged as administrator.

No error message is shown in this situation because this would visibly indicate the exact cause of a security violation to a potentially insecure user.

I do my unit and suite testing in Design mode. Is this a good practice?

Definitely not. When working as a designer additional menu options appear, security is disabled and additional features that use more resources are enabled.

When doing unit or suite testing you should always use VL Framework - as User entry point so that you experience the same environment that a real end-user would.

When I run the Framework security does not seem to work?

Are you using the VL Framework - as Designer entry point? If you are then refer to [I do my unit and suite testing in Design mode. Is this a good practice?](#)

Why options I have disabled for Windows and/or the Web browser are displayed?

Are you using the Framework as a designer? If you are then refer to [I do my unit and suite testing in Design mode. Is this a good practice?](#)

How can I translate text, for example the caption of the instance list Clear List button?

Refer to source code of the Imbedded Interface Point RDML function UFU0003 in process UF_SYSBR.

How do I enroll my own visual styles?

Refer to the source code of shipped component UF_SYSTM for more details.

Before using visual styles, investigate the use of a [Overall Theme](#). You may be able to get the appearance you want with fewer changes.

How do I change a visual style at run time?

If you need a visual style to change at run time, you can swap in a new style or styles from any command handler, filter, or snap-in instance list using this logic:

```
#avFrameworkManager.avSubstituteVisualStyle Ustyle(#MYSTYLE_A)  
Uasname('VF_VS106')  
#avFrameworkManager.avSubstituteVisualStyle Ustyle(#MYSTYLE_B)  
Uasname('VF_VS101') Usignalchanged(True)
```

Where `uStyle(#MYSTYLE_A)` is a visual style that you have defined in the repository, and `'VF_VS106'` is the name of one of the styles currently being used by the Framework.

If `uSignalChanged` is true, the Framework will apply the changes and the replacement styles will be visible.

For more information, see [Change a visual style at run time](#).

I have created a new command but it does not appear in the menus or the toolbar?

Just creating the command is not enough to make it visible. Commands only become visible in pop-up menus, toolbars or on menus when they are enabled within (i.e.: associated with) an object (the Framework, an application, a business object or a business object instance).

To add a command to a menu in the menu bar, use the Menus... or Commands... option of the Framework menu.

To enable a command double-click on the object and select the Commands Enabled tab.

Can I define the toolbar by application?

No, but a single toolbar button (and a command) can be used by many different applications and business objects. Toolbar buttons are reused (or shared) by applications, which is much the same thing.

Take the command "new" for example:

- You can make it appear on the toolbar by altering the definition of the command.
- You can enable the new command in applications 1, 2 and 3.
- When the button is clicked, the command handler in the current application (either application 1, 2 or 3) will be invoked to handle the command.

This way your users get used to using the single "New" button and it means "Make a new one of the thing I am currently working with".

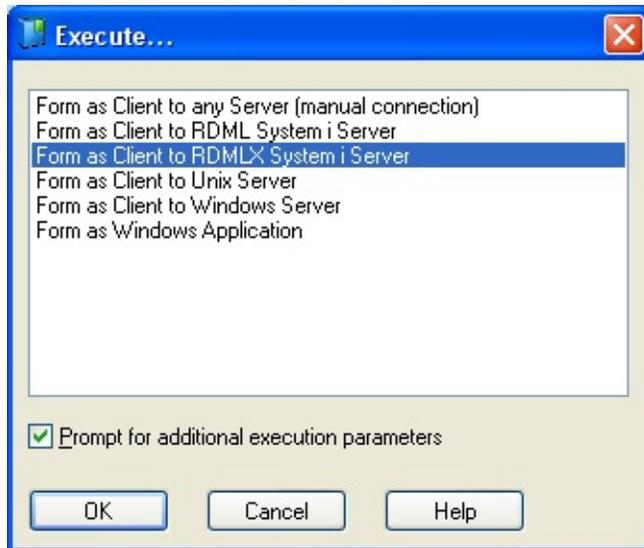
I change Instance Command Presentation but the changes do not take effect?

Sometimes you need to shut down and restart the Framework before complex changes (like using a separate window) will take full effect.

How do I run my application in SuperServer mode?

There are two ways of doing this:

1. Start the Framework running in SuperServer mode externally. You can do this by using the LANSAs folder icon titled Exec Form (to RDML System i) or Exec Form (to RDMLX System i), or by using the option to execute your form and choosing one of the "Client to xxxxxx Server" options.



Any of these options will establish your SuperServer connection externally (i.e.: before the Framework actually starts to execute). When using this method to start a SuperServer connection you should only execute UF_ADMIN or UF_EXEC. These will present your Framework exactly as a real end-user will see it. You cannot do Framework design activities while working this way.

2. Allow the Framework to make SuperServer connections itself.

By defining your server system within the Framework and setting the appropriate connection properties in the Framework design (see Administration/Servers) you can use UF_DESGN, UF_EXEC and UF_ADMIN in SuperServer mode.

The advantages of this method are:

- You can execute UF_DESGN this way, thus you can do design and command handler and filter testing activities while in SuperServer mode.
- The Framework can manage connection details (eg: user and password verifications, selective offline usage) for you,

- Your server definitions can be deployed.
- This environment more closely resembles what you would do in a real Framework that has been deployed to end-users.

Are copies of my Framework design kept?

If the [Keep XML File Versions](#) option in your Framework is turned on then back up copies of your Framework design are kept automatically.

A copy of your Framework design is created just before an altered Framework design is saved. In other words a snapshot of the current design is saved just before the altered design replaces it.

Your Framework design is normally saved:

- Every time you shut down the Framework if a change has been made to the design.
- Just before you snap in a real filter or command handler if a change has been made to the design. This is done in case your filter or command handler fails, causing the entire Framework to shut down abnormally.
- Every 10 minutes or whatever [Automatic Save Time in Minutes](#) interval is specified in your Framework, if a change has been made to the design, since the last time it was saved.

Normally Frameworks are stored in XML formatted files named VF_SY001_System.xml, that reside in the execute directory of the associated LANSa partition.

You will also find files with names like vf_Sy001_System_YYYYMMDD_HHMMSS.xml. These are the saved versions of your Framework design. The YYYYMMDD_HHMMSS portion of the name reflects the date and time they were saved.

If you copied file vf_Sy001_System_20010203_143000.xml over the top of the current VF_SY001_System.xml file then you would have reverted your Framework design (but not necessarily your Images Palette) back to as it was on the 3rd February, 2001 at 2:30pm.

I have just deleted an object but it is still visible?

Deleted objects are immediately marked as deleted. When you close the Framework and save the changes the object is deleted.

I have just deleted an object and want to get it back again?

To restore a deleted object you need to revert your entire Framework definition to the latest saved version that contained the deleted object. Refer to [Are copies of my Framework design kept?](#) for more details on how Framework versions are saved.

How do I change Help About logos?

The Framework and Application Help About Logos can be changed to any Framework enrolled bitmap. See [How do I enrol my own bitmaps and icons?](#)

To change the Help About Logo for the Framework, choose the Framework menu, and the Properties... option. Go to the Help About tab sheet. A very small version of your enrolled image should be visible in the Bitmaps group box at the bottom of the tab sheet. Select it.

To change the Help About Logo for an application, double-click on the application's icon and go to the Help About tab sheet. A very small version of your enrolled image should be visible in the "Bitmaps" group box at the bottom of the tab sheet. Select it.

How do I change introduction logos?

If you wish to change the default Framework and application images to an image of your own, the simplest method is to rename your image to UF_IM001.Gif and put it in the execute directory of the partition that the Framework is being run in. (replacing the shipped UF_IM001.Gif)

You can change the introduction images of the Framework and its applications individually. Firstly, add the image files you want to use to the execute directory of the partition that the Framework is being run in.

To change the Framework introduction image, choose the Framework menu, and then the Properties... option. Go to the Startup tab sheet. Change [Image File](#) to the name of the bitmap or gif that you want to use. (**Note:** Whether the image is displayed or not depends on the [Options](#) field, above the image file. If the option chosen is full image or real image, your new image should be displayed).

To change an application's introduction image, double-click on the application's icon and go to the Startup tab sheet. Change [Image File](#) to the name of the bitmap or gif that you want to use.

When deploying the Framework to your users, remember that you must deploy all of these image files to the user's partition execute directory.

How do I change the logo shown when the Framework starts executing?

Open the component UF_DESGN and follow the instructions in it.

I have just changed my Framework design but the change has not taken effect?

Most changes in the Framework are displayed instantly. In some cases with less commonly used options it may be necessary to shut down and restart the Framework to see a change.

How do I enrol my own bitmaps and icons?

Open component UF_IB001 and read the instructions.

Note: We will supply the shipped LANSVA VF_* bitmaps and icons on request.
Please contact product support for details.

Why are some menus and menu options in brackets?

See [Menu Options in Brackets](#)

How are Framework settings remembered?

The Framework remembers things between executions (eg: the main Framework window size, location and layout).

To find this information search for files with names like ppp_User_Virtual_ClipBoard.Dat and ppp_Framework_Virtual_ClipBoard.Dat stored in a temporary directory on your system (where "ppp" is the partition identifier). These files store virtual clipboard details between Framework executions.

They can be deleted at any time to clear the Framework's current virtual clipboard content. You may want to review [The Virtual Clipboard](#).

How are instance lists remembered?

In Windows applications instance lists are remembered. To find out where they are kept first find out the "Internal Identifier" of the business object involved, say, BUSINESSOBJ_N.

Then search for files with names BUSINESSOBJ_N_Part_1.Lst and BUSINESSOBJ_N_Part_2.Lst in a temporary directory on your system.

These files store the instance list details of BUSINESSOBJ_N between Framework executions. They can be deleted at any time to cause remembered instance list details to be lost. You should always delete both files (Part_1 and Part_2) together.

Can more than one person design the same Framework at the same time?

Yes. See [Development Architecture](#).

Where does the Images Palette get the pictures from (when using the HTML formatted Images Palette)?

It displays all the .GIFs files in the current partition's /EXECUTE directory.

To include a new GIF file into the palette simply copy it into the current partition's /EXECUTE directory.

When I execute my Framework in a browser some of my icons and bitmaps don't display properly?

The shipped Framework icons should display correctly when displayed in a browser. The small bitmaps for applications and business objects are not displayed in the web version of the Framework. If you have added your own icons, you need to create two equivalent gif images for each of these icons. One gif must be 16x16 pixels, and one must be 32x32 pixels.

How are icons and bitmap names used when my Framework is executed in a browser?

The web version of the Framework takes the icon name <name> and when displaying a small (16x16) image of an icon it looks for a gif called <name>1.gif.

When displaying a large (32x32) image of an icon it looks for a gif called <name>3.gif.

For large images that are not icons (for example, help about images) it simply looks for <name>.gif

For example, if you had enrolled an icon named MYICON into the Framework for use in Windows environments and you then used the icon in a Web Browser the Framework would look for MYICON1.GIF (16 x 16 small version) and MYICON3.GIF (32 x 32 large version).

How can I convert icons and bitmaps to GIF files?

MS Paint will convert bitmaps to gifs. When creating a gif, ensure that the color that represents transparent is known to the image editor. (in MS Paint this is specified under image attributes, if the image is a gif.)

To work with icons it is necessary to download an icon editor. These can be found for various prices at Tucows.com. Depending on the editor, you can either load the .ico file or save its images directly as gifs, or for cheaper icon editors you can load the icon and copy and paste it into MS Paint. Icon Studio 1.0 is an example of a freeware icon editor

Where do I need to put my GIFs so that my browser applications can use them?

While developing applications you need to put them into your private working folder and also into the LANSAs for the Web images directory. Eventually they will need to be deployed with your application.

Can my Web browser applications be used with System i multi-tier web server configurations?

Yes, but there are some additional configuration things you need to consider.

The browser version of the Visual LANSA Framework generates pages at runtime that are included in the LANSA for the Web page via Server Side Includes (SSI).

In a multi tier environment the web server and the application server are on different machines. (This is sometimes called a Model B implementation.)

Hence there is a requirement for the application server (where the include page is generated) to be able to have access to the web servers IFS system.

There is an IBM supplied file server called 'QFileSvr.400'. Creating a directory in there using the host name of another machine in the network gives you access to the other machine's IFS.

Conditions:

- The System i machines must be in the same network.
- The user profile and password used by the LWEB job must exist in the web server machine.
- The host directory created under QFileSvr.400 is NOT persistent across IPLs. It must be recreated after each IPL.
- A firewall between machines might require extra configuration steps.

Command to execute:

```
MKDIR DIR('/QFileSvr.400/<target System i host name>')
```

Can my Web browser applications be used with Windows multi-tier web server configurations?

The browser version of the Visual LANSA Framework generates pages at runtime that are included in the LANSA for the Web page via Server Side Includes (SSI).

There is a requirement for the application server (where the include page is generated) to be able to access the web server's file system. Whether the application server is a System i or a Windows PC a virtual directory must be created in the Windows web server.

If the application server is a System i server

The easiest way to configure the temporary files directory is to use a directory in the System i IFS system. In this example the directory nominated for the temporary files is **/tmp** and the virtual directory (alias) created in the Windows web server is **VLF_Temp**.

Step 1. On the Windows web server PC, open the Internet Information Services located in the Administrative Tools folder in the Control Panel.

Step 2. Right click on the Default Web Site, select New and then Virtual Directory.

Step 3. Follow the instructions in the Wizard:

Alias	Visual LANSA Framework_Temp
Directory	Use the Browse button to locate the tmp directory on your System i IFS. Note that you must have mapped a drive to /tmp
Access Permissions	Allow Directory Browsing
User Id and Password	Specify the user and password to access the IFS directory

Step 4. Stop and Restart the IIS Admin Service located in the Administrative Tools folder in the Control Panel.

Step 5. Verify the virtual directory is accessible. Start Internet Explorer and type http://localhost/vlf_temp/

Execute the Administrator's Console:

```
http://<your host>/cgi-bin/lansaweb?procfun+vf_pr004+vf_u0402+<ppp>
```

Where <your host> is the IP address of your LANSAs for the Web server and <ppp> is the partition to be used.

The values for the Real and Virtual Directories would then be set like this:

The screenshot shows a 'System Settings' window with the following configuration:

- Is the Framework Available?** Yes No
- Message when framework not available** Framework not available
- Real directory for temporary files** /LANSA_vl_xpgmlib/webserver/images/vlf_temporary_files/
- Default value for your LANSAs system** /LANSA_VLXPGMLIB/webserver/images/vlf_temporary_files/
- Virtual directory for temporary files** /images/vlf_temporary_files/
- Default value for your LANSAs system** /images/vlf_temporary_files/

If the application server is a Windows PC

Conceptually the issues are the same. However, if you use images or a subdirectory of it, you might not require the configuration of a virtual directory. In this example:

- The directory nominated for the temporary files on the Web Server machine is c:\Lansa\WebServer\Images\VLF_Temp\.
- On the application server, drive letter **G** is mapped to the Web Server machine's c:\Lansa\WebServer\.

The values for the Real and Virtual Directories would then be set like this:

System Settings

Is the Framework Available? Yes No

Message when framework not available

Real directory for temporary files

Default value for your LANSa system

Virtual directory for temporary files

Default value for your LANSa system

How can I change the visual styles used in Web browser applications

The Framework uses Cascading Style Sheets to define the visual aspects of Web browser applications. The cascading style sheets are shipped in three sets where each set supports a different skin. The shipped cascading style sheets are:

Cascading Style Sheet Name	Description	Should you modify this?
VF_VS001.css	Styles common to all skins.	No.
VF_VS001_<skin>.css where <skin> is WIN, WEB or XP	Styles specific to each skin that you should not modify.	Not recommended.
UF_VS001_<skin>.css where <skin> is WIN, WEB or XP	Styles specific to each skin that you might modify.	Possibly.

Please note the following before attempting to modify any style sheet:

- New versions of the Framework will overwrite any installed cascading style sheet files. You need to save any changes you make and may need to reapply them after upgrading to new Framework versions.
- You need some knowledge of cascading style sheets to make style sheet changes. Numerous training and tutorial resources related to cascading style sheets can be found on the Internet.
- Support issues related to changes you make to shipped style sheets will not be accepted.
- Style sheets are complex to implement and test. They may represent a significant diversion that impacts on your project schedules. We recommend making style sheet changes after you have completed implementing the functional components of your application.

What is a temporary directory and what is it used for?

The temporary directory, which is defined via the Administrator's console, is used to store temporary state information for active Web browser Visual LANSAs Framework applications.

It is also used to dynamically create XML data islands that are inserted into web pages being back sent to client browser applications.

With regard to the location of the temporary directory:

- The temporary directory used can be anywhere in a network, but it is best physically located on the LANSAs application server for performance reasons.
- The LANSAs application server uses the real temporary directory name when it is creating the state management and XML data island files. It needs this name to be correctly specified (from it's point of view) so that it can write to files in this directory.
- The LANSAs application server never directly uses the virtual temporary directory name. However, it needs to know what it's name is (from the HTTP server's point of view) so that it can generate HTML pages containing correctly formatted `<!--#include virtual="xxxxxxxxxxxxx" -->` directives for the HTTP server. These are used to dynamically insert the XML data islands into web pages being sent back to client browsers.
- The HTTP server needs to be able to locate the temporary directory by using the virtual name specified in the `<!--#include virtual="xxxxxxxxxxxxx" -->` directives. This allows it to read the specified file and include it into the HTML page that is sent out to the client's browser. This is why SSI (server side includes) need to be enabled in your chosen HTTP server.

How can I purge old information from my temporary directory?

Normally temporary files are deleted automatically when web users sign off or close their web browser, but over time old temporary files may collect in the temporary directory (eg: turning your PC off while using a browser application would leave its temporary files in the temporary directory).

To purge old temporary files use the Administrator's Console.

There are options available to purge all temporary directory files or only files that are more than 2 days old. You should not use the option to purge all files while people are using browser applications.

Can I purge old information from my temporary directory in a batch job?

Yes. The purging of the temporary directory can be done by calling LANSAPUR function VFU0030.

In the example VFU0030 is being called to purge data 4 or more days old:

```
Define Field(#DAY_RANGE) Type(*DEC) Length(001) Decimals(0)
Change #Day_Range 4
Exchange Fields(#DAY_RANGE)
Call Process(*DIRECT) Function(VFU0030) Exit_Used(*NEXT) Menu_Used
```

How can users sign off from VLF-WEB sessions?

You close or sign off a Web browser application in two ways:

- By using a Signoff or Exit VLF command initiated from for example a button.
- By closing the web browser window (for example by Using the red "x" option at the top right of the browser window).

When you use the signoff /exit command option, a message is sent back to the web server indicating that you are signing off. The web server then deletes any temporary files that you have in use.

When you use the "x" option to close the browser window, it is not always possible to send the signoff message back to the web server because your main web browser window is in the process of closing. This behavior depends on the browser you are using. If the message is not sent, any temporary files associated with your session will be left in place until they are purged by the next folder clean up you run.

Text appears in English when I execute Framework as a web application

The Visual LANSA Framework browser extension uses a number of multilingual text strings to issue status or error messages, as captions, etc. By default these text strings appear in English. To translate the shipped English text strings:

- Locate the multilingual source file **VF_MULTI_ENG.JS** in the execute directory of the partition where you have imported the Visual LANSA Framework.
- Copy it to another file and rename the new file as **VF_MULTI_LLLL.JS** where LLLL is the desired language code (e.g. FRA, TCHI, etc.).
- Edit the newly created **VF_MULTI_LLLL.JS** with any text editor (e.g. Notepad) and follow the instructions provided in the source.
- Save the file with the new translations.

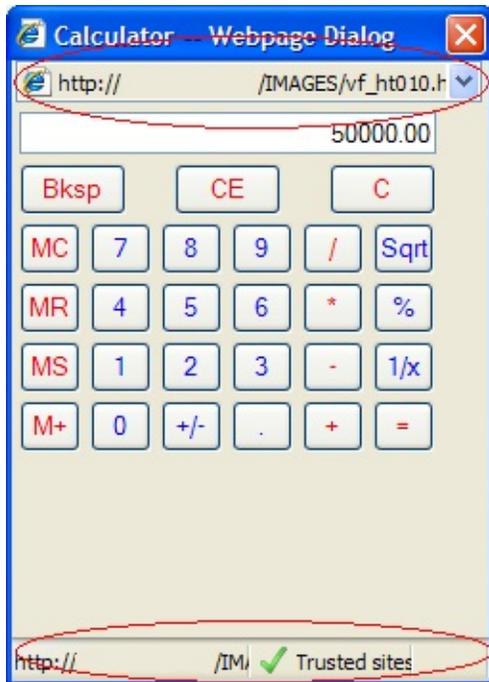
I want to restore the default window layout for the Framework

See [How are Framework settings remembered?](#)

How can I hide the address and status bars on Framework popup windows when using IE7?

Internet Explorer version 7 has security features that, with default settings, prevents the Visual LANSAs Framework from hiding the Address and Status bars on popup windows.

For example a popup window with a calculator can appear like this:



To prevent the address and status bars appearing on pop-up windows in your Web browser applications perform the following steps:

- Close any popup windows displaying unwanted address or status bars.
- Open Internet Options from the Tools menu.
- Select the Security tab.
- Select the security zone for your LANSAs for the Web server. Note: It is recommended that you list your LANSAs for the Web server as a Trusted Site.
- Select the Custom Level button.
- Scroll down to the Miscellaneous group of settings.
- Under 'Allow web sites to open windows without address or status bars' select the Enable radio button.
- Click OK to close Security Settings.

- Click Yes on the change of settings for this zone warning.
- Click OK to close Internet Options.
- Open required pop-up windows and address and status bars should not be displayed.

Note: Customizing security settings for zones other than the Trusted Sites zone can pose serious security risks and is not recommended unless you are certain of the security implications.

The URL to start my deployed VLF web browser application is too complex for users to reliably type in to their browsers

Imagine that the URL needed to start your deployed web browser application was:

```
http://nnn.nn.nn.nn/VF_SY001_System_ENG_BASE.HTM?
Partition=PRO+SwitchTo=xxxx+SwitchObject=xxxxx+SwitchCommand=xxxxx
```

This is too complex for anyone to reliably type in. Some of the options you have to avoid having people type it in to a browser are:

Option 1: Create a short cut on their desktop that resolves to the URL.

You can even create the shortcut on your desktop, then drag and drop it into an e-mail. The e-mail recipients can then drag and drop it from the e-mail onto their desktops. When they click on the shortcut on their desktop the web browser will be started.

Option 2: Create a simplified start up HTML file.

If you create an HTML file named MyApplication.htm (say) on your web server that contains HTML like this:

```
<html>
<head><title>My Application</title></head>
<body onload='window.location.replace(
"http://nnn.nn.nn.nn/VF_SY001_System_*_BASE.HTM?
Partition=PRO+SwitchTo=xxxx+SwitchObject=xxxxx+SwitchCommand=xxx
);'>
</body>
</html>
```

Then the users only ever need to type in <http://nnn.nn.nn.nn/MyApplication.htm> to start the application. An additional benefit is that you can change the parameters used in MyApplication.htm at any time without the users needing to change what they do. Of course you can still use option 1 to avoid typing in any URL at all.

What should I do when I get a mysterious WAM crash in my Framework RAMP application?

Scan your drives for these files and delete all occurrences of them:

- X_ERR.LOG
- LANSAWEB.LOG
- LX_WAM.LOG

Then execute the WAM application again to the point of failure and then scan for these files again across your whole hard drive.

One or more of them will most likely contain useful error information.

How do I pick up the servers that have been defined in the Framework using the Servers option of the Administration menu?

This code snippet demonstrates how to display a message showing all the servers defined in the network:

```
* Use VL's F2 feature help to see all properties exposed by this server object
```

```
Define_Com Class(#VF_FP007) Name(#Server) Reference(*dynamic)
```

```
* Get the first server
```

```
Invoke Method(#uSystem.uServerContainer.uFirstServer) Userverref(#Server)  
Ucursor(#vf_elnum) Ureturncode(#vf_elretc)  
Dowhile Cond('#vf_elretc = ok')
```

```
* The VF_FP007 server object has a set of properties that define the server.
```

```
* These properties are READ ONLY in this context and must not ever be  
updated.
```

```
* For example ..
```

```
Use message_box_add ('Caption' #Server.uCaption)  
Use message_box_add ('Type' #Server.uServerType)  
Use message_box_add ('LU Name' #Server.uServerLUName)  
Use Message_box_show (ok ok info)
```

```
* Get the next server
```

```
Invoke Method(#uSystem.uServerContainer.uNextServer)  
Userverref(#Server) Ucursor(#vf_elnum) Ureturncode(#vf_elretc)  
Endwhile
```

Why do I get a message saying 'Compression source file vf_multi_NAT.js not found'?

This happens when saving the Framework in a monolingual partition.

When the Framework runs in the browser it uses a file named vf_multi_LLL.js (where LLL is the language code) to access multilingual text string values. There are three such files shipped: vf_multi_ENG.js, vf_multi_FRA.js and vf_multi_JPN.js.

If the language of your monolingual partition is English, French or Japanese, simply copy the file corresponding to your language with the new name of vf_multi_NAT.js. These files are located in the partition execute folder.

If the language of your monolingual partition is none of the above, copy the file but also edit it and modify it according to your own language.

Why do I get a message saying 'Compression source file vf_multi_LLL.js not found'?

This happens when saving the Framework in a multilingual partition.

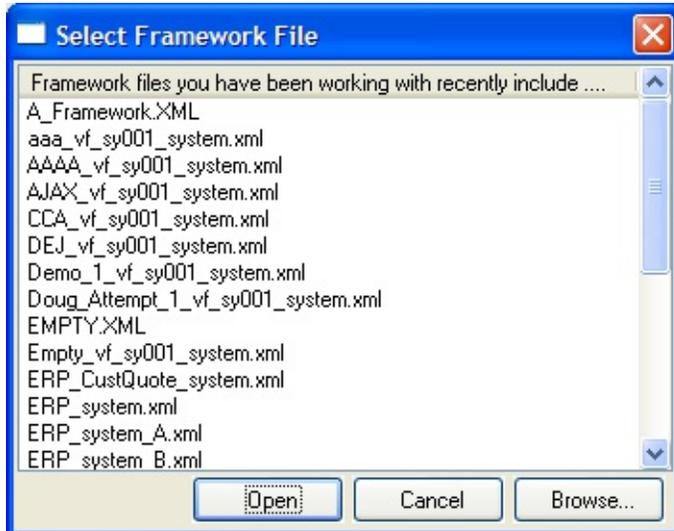
When the Framework runs in the browser it uses a file named vf_multi_LLL.js (where LLL is a language code) to access multilingual text string values. There are 3 such files shipped: vf_multi_ENG.js, vf_multi_FRA.js and vf_multi_JPN.js.

Simply copy one of those files with the new name of vf_multi_LLL.js where LLL is the desired language code. These files are located in the partition execute folder.

You will then have to edit and modify the contents to reflect your own language.

How can I change the list of Framework versions shown?

If you have a lot of Frameworks and start up as a VLF designer you are asked to choose which one to use:



If you want to change this list of choices, or keep different lists of choices, then it's worth knowing:

- The displayed list of choices comes from a text file that you name in your VLF system entry point.
- The shipped entry point UF_DESGN uses the name `vf_sy001_system_choice.txt` for this file because of this line of code:

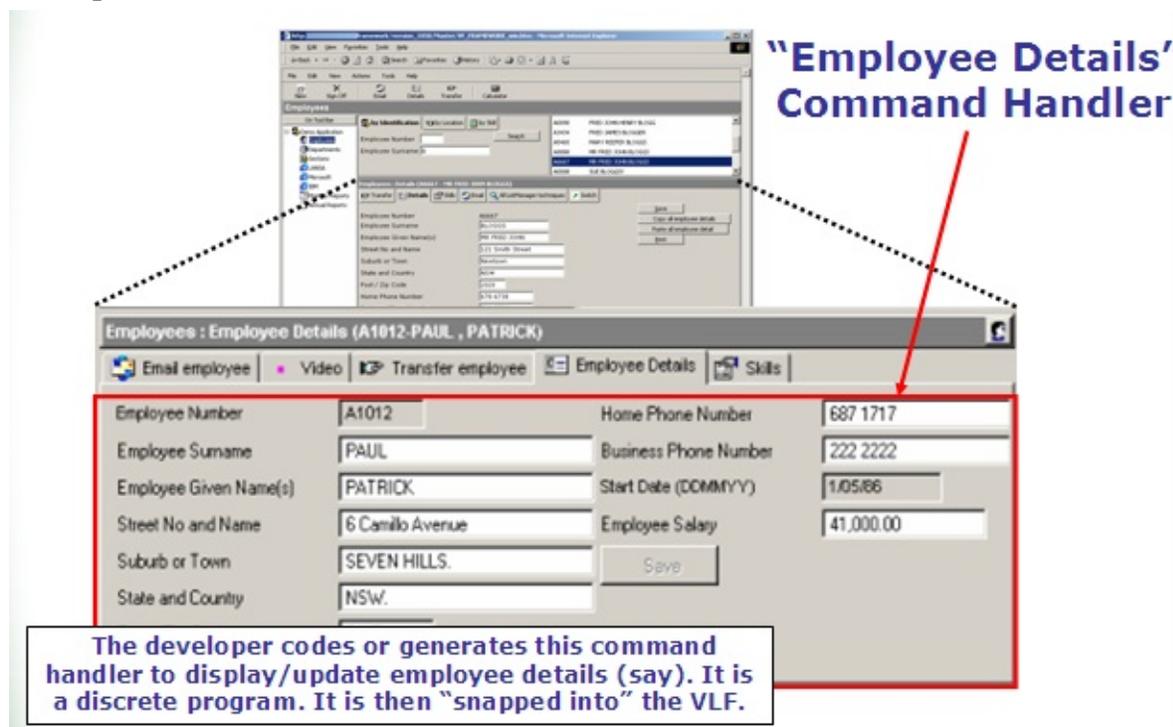
```
Set Com(#Com_Owner) Usystemxmlfile('vf_sy001_system.xml')  
Usystemxmlchoice('vf_sy001_system_choice')
```

- `vf_sy001_system_choice.txt` is kept in the LANSAsystem's partition execute folder, so it is shared on multi-developer systems.
- The last Framework you used is kept in a file named `vf_sy001_system_choice_Last_Used.txt`, which resides in your personal temporary directory (`*TEMP_DIR`) because it reflects what you used last.
-

Framework Programming

Read this section only when you are finished with your prototype.

You turn your prototype into a real application by writing programs that act as real filters and command handlers. As you complete each filter or command handler you snap it into the Framework, replacing the prototype version. For example:



Your Framework gradually evolves from being a prototype application into a real application as you replace each prototype filter and command handler with a real one.

Any filter or command handler program that you write must be one of these types:

WINDOWS A Visual LANSA component that works in the native Windows environment.

See [Windows Filter and Command Handler Anatomy](#)

WAM

A LANSA for the Web component that works with the web browser interface. It is coded as an RDMLX component that uses WEBROUTINE commands.

You can only use WAM command handlers and filters if you

have LANSAs 10.5 or later installed.

See [WAM Filter and Command Handler Anatomy](#)

Hidden command handlers are a type of command handler that perform tasks but which are not displayed. See [Hidden Command Handler Anatomy](#).

Remember that normally you don't have to code filters or command handlers by hand. The Framework comes with a [Program Coding Assistant](#) that will generate the code for you.

Once you understand the basic anatomy of filters and command handlers you can progress on to understanding the Framework facilities that you can use inside them:

- [Framework Ancestor Components \(WINDOWS only\)](#)
- [List Manager and Instance Lists](#)
- [Framework Manager](#)
- [Web Programming](#)
- [Designing Filter and Command Handler Layouts](#)

As you gain experience with filters and command handlers you might also be interested in some advanced features:

- [Advanced Filter Styles](#)
- [Custom Properties](#)

Also see:

- [Using Unicode Data with the Framework](#)
- [End-user Help \(F1\)](#)
- [Programming Tips](#)

Windows Filter and Command Handler Anatomy

Applies to: WINDOWS only.

WINDOWS filters and command handlers:

- Are used for all Windows applications.
- Are usually visual in nature
- Can have their forms painted by using the normal Visual LANSA form painter (i.e.: use the Design tab when reviewing the source code for the filter or command handler in the Visual LANSA editor).
- Execute on the client system
- Can maintain their state because they are Windows based.

Structurally WINDOWS filters or command handlers look like this:

```
1 Begin_Com Role(*EXTENDS #VF_ACxxx)
    Mthroutine Name(uInitialize) Options(*REDEFINE)
2     << your logic goes here >>
    Endroutine

    Mthroutine Name(uTerminate) Options(*REDEFINE)
3     << your logic goes here >>
    Endroutine

    Mthroutine Name(uExecute) Options(*REDEFINE)
4     << your logic goes here >>
    Endroutine

    Mthroutine Name(uActivate) Options(*REDEFINE)
5     << your logic goes here >>
    Endroutine

    Evtroutine Handling(#Save_Button.Click)
    << your logic goes here >>
    Endroutine
6
    Evtroutine Handling(#Transfer_Button.Click)
    << your logic goes here >>
    Endroutine

End_Com
```

There are several important parts in any filter or command handler:

- 1 All filters and command handlers extend (i.e.: inherit from) a base class shipped with the Framework. Filters extend a class named #VF_AC007 and command handlers extend a class named #VF_AC010. In both cases the base classes provide a set of pre-defined behavior to the command handler or filter.
- 2 All filters and command handlers can have an optional method named uInitialize that is executed just once when the filter or command handler is being created. Typically you use this to fill in details on the form that the filter or command handler will display.

- 3 All filters and command handlers can have an optional method named `uTerminate` that is executed just once when the filter or command handler is being destroyed. Typically you use this to free up things that the filter or command handler uses.
- 4 Command handlers normally have a `uExecute` method. This method is invoked whenever the user executes the Framework command that is associated with the command handler. Filters do not have `uExecute` methods because there is no command associated with them. They simply display their user interface and then wait for the user to indicate what they want to do (eg: Click the Search button).
- 5 Command handlers can have an optional method named `uActivate` that executes when the user causes the command handler to be redisplayed (ie: activated). Typically when switching back to a command handler from another business object – but they did not execute the command. Use of this method is quite rare and very specialized. Make sure you understand its purpose and invocation timing before using.
- 6 Most filters and command handlers also have their own unique event handling routines (`EvtRoutine/EndRoutine` command pairs) and methods `MthRoutine` (`Mthroutine/EndRoutine`) command pairs. This example shows "Click" event routines for two buttons named `#Save_Button` and `#Transfer_Button`. The logic inside these routines would define what happens when the respective buttons are clicked by the user.

Writing **WINDOWS** filters or command handlers is just like writing any other Visual LANSA reusable visual part, except that the Framework manages how, when and where your filter or command handler is created.

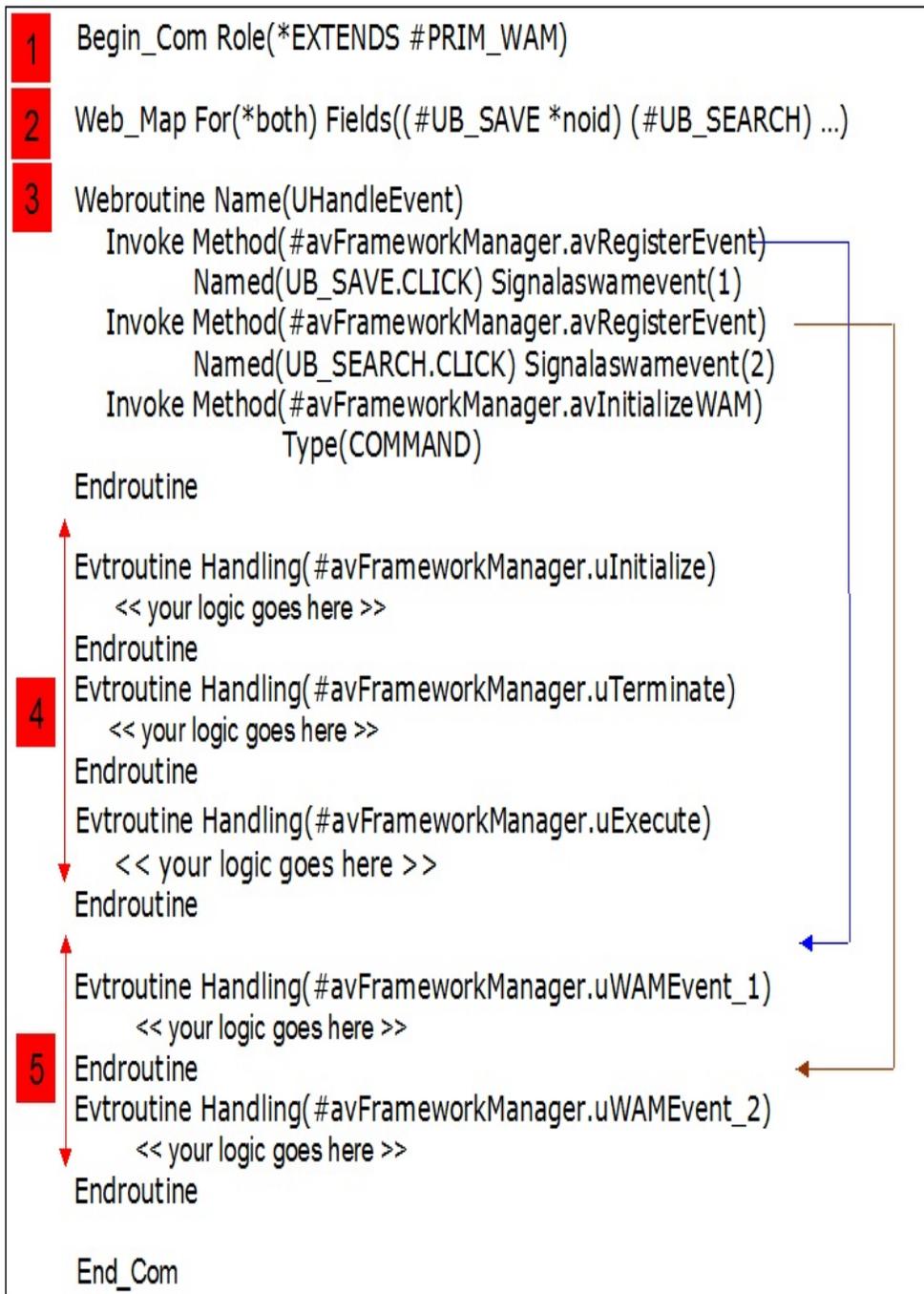
WAM Filter and Command Handler Anatomy

Applies to **WAM** only.

WAM filters and command handlers:

- Can only be used if you are using LANSAs 10.5 (or later)
- Are usually visual in nature
- Uses vlf_layout or vlf_layout_v2 as its layout weblet. The vlf_layout_v2 weblet is used to enable the use of Weblets with the jQuery UI design theme.
- Can have their forms painted by using the WAM editor.
- Execute on the server system
- Cannot maintain their state because they are web browser based

Structurally the basic anatomy of a **WAM** function looks like this:



There are five important things you need to understand about the structure of WAM filters and command handlers:

- 1 All WAM filters and command handlers extend (i.e.: inherit from) a base class named #PRIM_WAM. The base classes provide a set of pre-defined behavior to the WAM.
- 2 There is always a web map that defines what is sent out as the web page.

In this example the push buttons #UB_SAVE and #UB_SEARCH are sent.

3 There is always a single WEBROUTINE defined that does two important things:

– It registers all the events that the filter or command handler can handle and associates them with an event handling routine. In this example the click events for the #UB_SAVE and #UB_SEARCH buttons are registered and associated with event routines (the blue and brown arrows show the associations).

– It Indicates to the Framework manager that the Framework can be initialized and executed.

The only code you should ever add to the uHandleEvent routine is avRegisterEvent method invocations. All other logic should be put into the appropriate uInitialize, uExecute, uTerminate or uWAMEvent_N event handling routines.

4 Filters and command handlers can have optional uInitialize, uTerminate and uExecute event handlers. Since WAM routines are created and destroyed every time they are executed, the uInitialize and uTerminate routines are executed every time the WAM is invoked. Therefore they should only contain code that you want to execute at the start and/or end of every single WAM interaction with the client.

uExecute is only ever executed when the WAM is executed (that is, when a filter is started or a command handler is executed). When events occur inside an active WAM (for example a button click) uExecute is not signalled, just the registered uWAMEvent_N event.

5 Most filters and command handlers also have their own unique event handling routines. The example WAM used here has registered events (UB_SAVE.Click and UB_SEARCH.Click) so it has 2 event routines defined to handle the respective click events.

Routines execute in a WAM for VLF for Web in this order:

1. uHandleEvent (Note that messages from routine uHandleEvent will be sent last even though it executes first)
2. uInitialize
3. uExecute (first time logic)

4. User WAM Events
5. uTerminate

Writing **WAM** filters or command handlers is much like writing any other WAM program except that the Framework manages how, when and where the filter or command handler is created, displayed and destroyed.

Normally you generate your WAM by using the Program Coding Assistant initially and then when you need to add another event simply:

- Register the event and associate it with an event handling routine.
- Add in a new event handling routine to handle the event.

The most important thing to remember is that whenever your WAM filter or command completes handling an event it then **ceases to exist**.

Hidden Command Handler Anatomy

Applies to: **Windows** and **Web** applications.

Hidden command handlers:

- Are run in the same way as other commands but do not appear on tabs or in separate windows and are hidden from the user
- Are used to perform non-visual tasks.
- For **Windows** applications are reusable parts with `vf_ac020` set as their ancestor.
- For **Web** applications they are simply Visual LANSA functions.
- Have most non-visual Framework and instance list services available to them.

Structurally, Hidden Command Handlers for **Windows** applications are similar to **Windows** Command Handlers with these important differences;

- They extend the base class `#VF_AC020`.
- They don't use the optional method `uInitialize`.
- They don't use the optional method `uTerminate`.
- They don't listen to events.

Hidden Command Handlers for **Web** applications are:

- Normal Visual LANSA functions.
- They don't include visual elements such as Request or Display commands.
- They cannot signal to commands in the same business object at the same level.
- They do not listen to events.

When using hidden command handlers it is important to remember that they;

- Should always have the Default Command option set to NEVER for instance level commands or NO for business object level commands.
- Should never be used with the Hide All Other Command Tabs option set.
- Should never be attached to RAMP Destination screens.
- Should never attempt to display information to the user or interact with the user. Hidden means hidden.

Example of a Hidden Command Handler for Windows Applications
Example of a Hidden Command Handler for Web Applications

Example of a Hidden Command Handler for Windows Applications

```
* =====
* Type      : COMMAND HANDLER
* Platform  : MS-WINDOWS (Visual LANSA)
* Ancestor  : VF_AC020
* Copyright : (C) Copyright
* Framework : Dem Framework
* =====
* An instance level command, reverses the Employee Name in the instance list
* =====
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #VF_AC020)

* =====
* Simple Field and Group Definitions
* =====

DEFINE FIELD(#REVSD) REFFLD(#VF_ELBOOL)
DEF_COND NAME(*REVSD) COND('#REVSD *EQ TRUE')

* =====
* Handle Command Execution
* =====

Mthroutine Name(uExecute) Options(*REDEFINE)

* Do any execution logic defined in the ancestor

Invoke #Com_Ancessor.uExecute

* Get the Employee number of the current instance

Invoke #avListManager.GetCurrentInstance AKey1(#EMPNO) AColumn3(#R

* Fetch information from the main file to fill in the header fields on the form
```

FETCH FIELDS(#SURNAME #GIVENAME) FROM_FILE(PSLMST) WITH

* Put the names together in the reverse order

If Cond(*REVSD)

* Put the names together Given name first

Change #UF_VisID2 #GIVENAME

Use BConcat (#UF_VisID2 #SURNAME) (#UF_VisID2)

* Set the reversed flag

Change #REVSD FALSE

Else

* Put the names together Surname first

Change #UF_VisID2 #SURNAME

Use BConcat (#UF_VisID2 #GIVENAME) (#UF_VisID2)

* Set the reversed flag

Change #REVSD TRUE

Endif

* Update the name (Visual ID 2) to the instance list

Invoke Method(#avListManager.UpdateListEntryData) AKey1(#EMPNO) Vis

Endroutine

End_Com

Example of a Hidden Command Handler for Web Applications

Example 1

- * This is an example of VLF.WEB hidden command.
- * It is executed on the web server.
- * It can coded in RDML or RDMLX format.

- * Regardless of whether it is RDML or RDMLX it needs to use the
- * VF* series of built-in functions to communicate with the Framework
- * manager.

Function Options(*DIRECT)

- * Get the current instance list entry

Use Builtin(VF) With_Args(GETCURRENTINSTANCE)

- * Get AKEY3 because it contains the employee number

Use Builtin(VF_GET) With_Args(AKEY3) To_Get(#EmpNo)

- * Issue a message showing the employee number retrieved

Execute Subroutine(Showmsg) With_Parms('Employeee number selected is'
#Empno)

- * Make the message always be displayed in the web browser

Use Builtin(VF_SET) With_Args(AVSHOWMESSAGES TRUE)

- * Finished

Return

- * Simple subroutine to issue a message onto program message queue

Subroutine Name(ShowMSG) Parms((#MSGDTA *RECEIVED) (#TEMP1
*RECEIVED))

Define Field(#Msgdta) Type(*char) Length(132)

```
Define Field(#Temp1) Type(*char) Length(132)
Use Builtin(BCONCAT) With_Args(#MSGDTA #TEMP1)
To_Get(#MSGDTA)
Message Msgid(DCM9899) Msgf(DC@M01) Msgdta(#MsgDta)
Endroutine
```

Example 2

```
* =====
* Description ...: Instance level Hidden Command Handler
*
* =====
FUNCTION OPTIONS(*DIRECT)

* Simple Field and Group Definitions
* =====
DEFINE FIELD(#REVSD) REFFLD(#VF_ELBOOL)
DEF_COND NAME(*REVSD) COND('#REVSD *EQ TRUE')

* Get the Employee number of the current instance
* and the reversed flag.

USE BUILTIN(VF) WITH_ARGS(GETCURRENTINSTANCE)

USE BUILTIN(VF_GET) WITH_ARGS(AKEY1) TO_GET(#EMPNO)

* Get the Name state from the clipboard

USE BUILTIN(VF_RESTOREAVALUE) WITH_ARGS(*BLANKS PNCAFL

USE BUILTIN(VF_TRACEAVALUE) WITH_ARGS('Employee name is surn

* Fetch information from the main file to fill in the
* header fields on the form

FETCH FIELDS(#SURNAME #GIVENAME) FROM_FILE(PSLMST) WITH

* Put the names together in the reverse order
```

```
IF COND(*REVSD)
```

```
* Put the names together Given name first
```

```
CHANGE FIELD(#UF_VISID2) TO(#GIVENAME)
```

```
USE BUILTIN(BCONCAT) WITH_ARGS(#UF_VISID2 #SURNAME) TO_C
```

```
* Set the reversed flag
```

```
CHANGE FIELD(#REVSD) TO(FALSE)
```

```
USE BUILTIN(VF_TRACEAVALUE) WITH_ARGS('Employee name has be
```

```
ELSE
```

```
* Put the names together Surname first
```

```
CHANGE FIELD(#UF_VISID2) TO(#SURNAME)
```

```
USE BUILTIN(BCONCAT) WITH_ARGS(#UF_VISID2 #GIVENAME) TO_
```

```
* Set the reversed flag
```

```
CHANGE FIELD(#REVSD) TO(TRUE)
```

```
USE BUILTIN(VF_TRACEAVALUE) WITH_ARGS('Employee name is now
```

```
ENDIF
```

```
* Update the name (Visual ID 2) to the instance list
```

```
USE BUILTIN(VF) WITH_ARGS(BEGINLISTUPDATE)
```

```
* Set up the list entry details
```

```
USE BUILTIN(VF_SET) WITH_ARGS(VISUALID1 #EMPNO VISUALID2
```

```
USE BUILTIN(VF_TRACEAVALUE) WITH_ARGS('Employee name is surn
```

```
* Add instance details to the instance list
```

USE BUILTIN(VF) WITH_ARGS(ADDTOLIST)

* Instance list updating is now complete

USE BUILTIN(VF) WITH_ARGS(ENDLISTUPDATE)

* Remember which way the name is presented

USE BUILTIN(VF_SAVEAVALUE) WITH_ARGS(#REVSD PNCAFLT REV

Framework Ancestor Components

Applies to WINDOWS only.

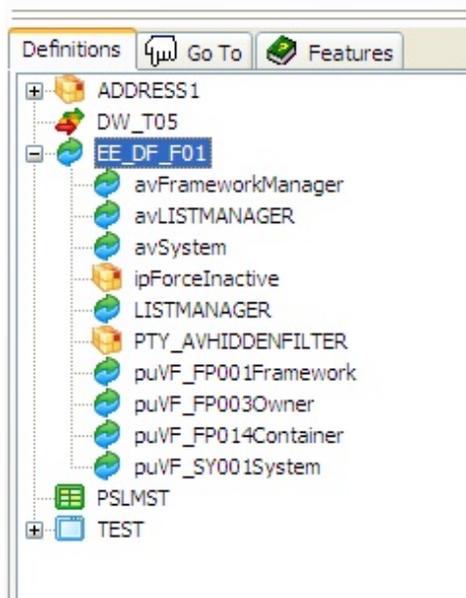
If you are creating a filter, command handler or instance list browser you need to use a standard shipped ancestor. This means that your component will inherit some predefined behavior, thus minimizing the amount of code you need to write.

When you create a Reusable Part that will be used as a filter, command handler or instance list, you need to set the Role property appropriately:

Type of Reusable Part: The code at beginning of your program must say:

Filter	Begin_Com Role(*EXTENDS #VF_AC007) ... etc ,,,
Command Handler	Begin_Com Role(*EXTENDS #VF_AC010) ... etc ,,,
Instance List browser	Begin_Com Role(*EXTENDS #VF_AC012) ... etc ,,,
Hidden Command Handler	Begin_Com Role(*EXTENDS #VF_AC020) ... etc ,,,

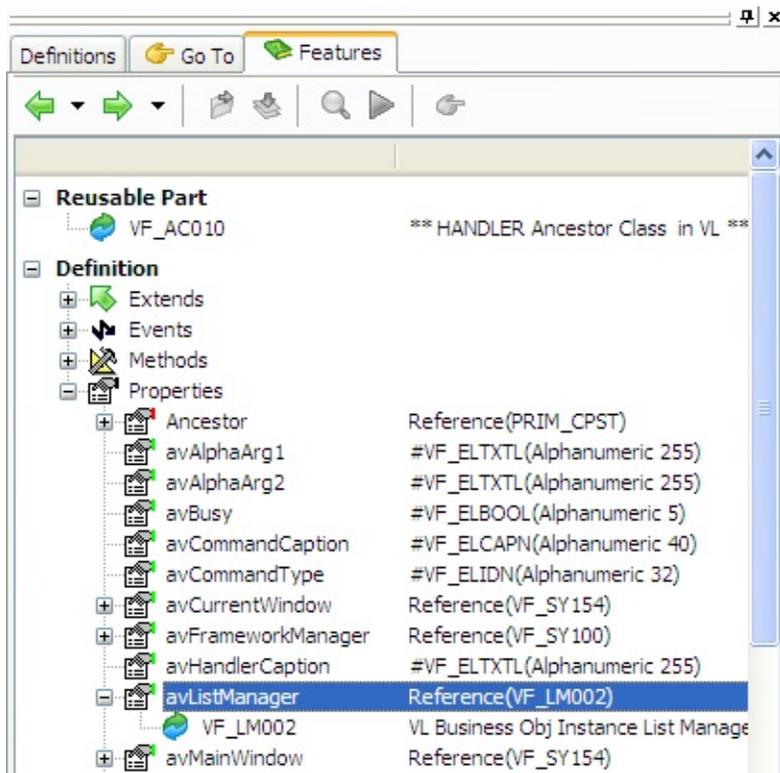
Filters, command handlers and instance lists make use of services provided by two components, [List Manager and Instance Lists](#) and [Framework Manager](#):



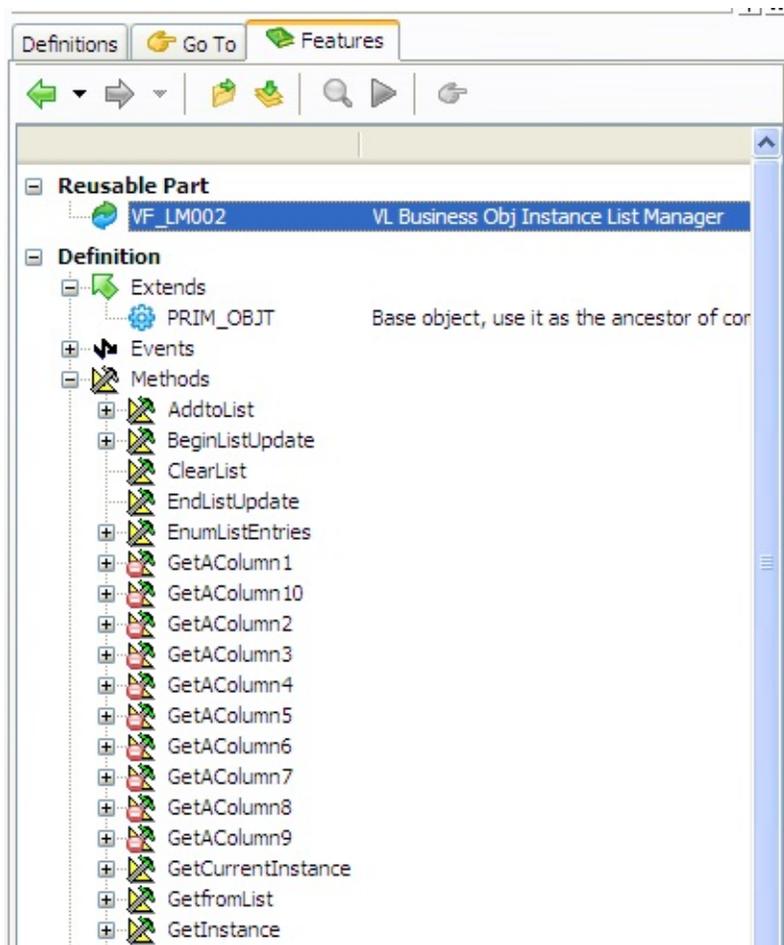
(Ignore the PUVF_ components, they are internal.)

The best way to explore the List Manager and the Framework Manager services

is to select avListManager or avFrameworkManager in the Visual LANSA outliner:



Double-click them and then use the component property sheets and F2 to investigate the properties, methods and events that they provide:

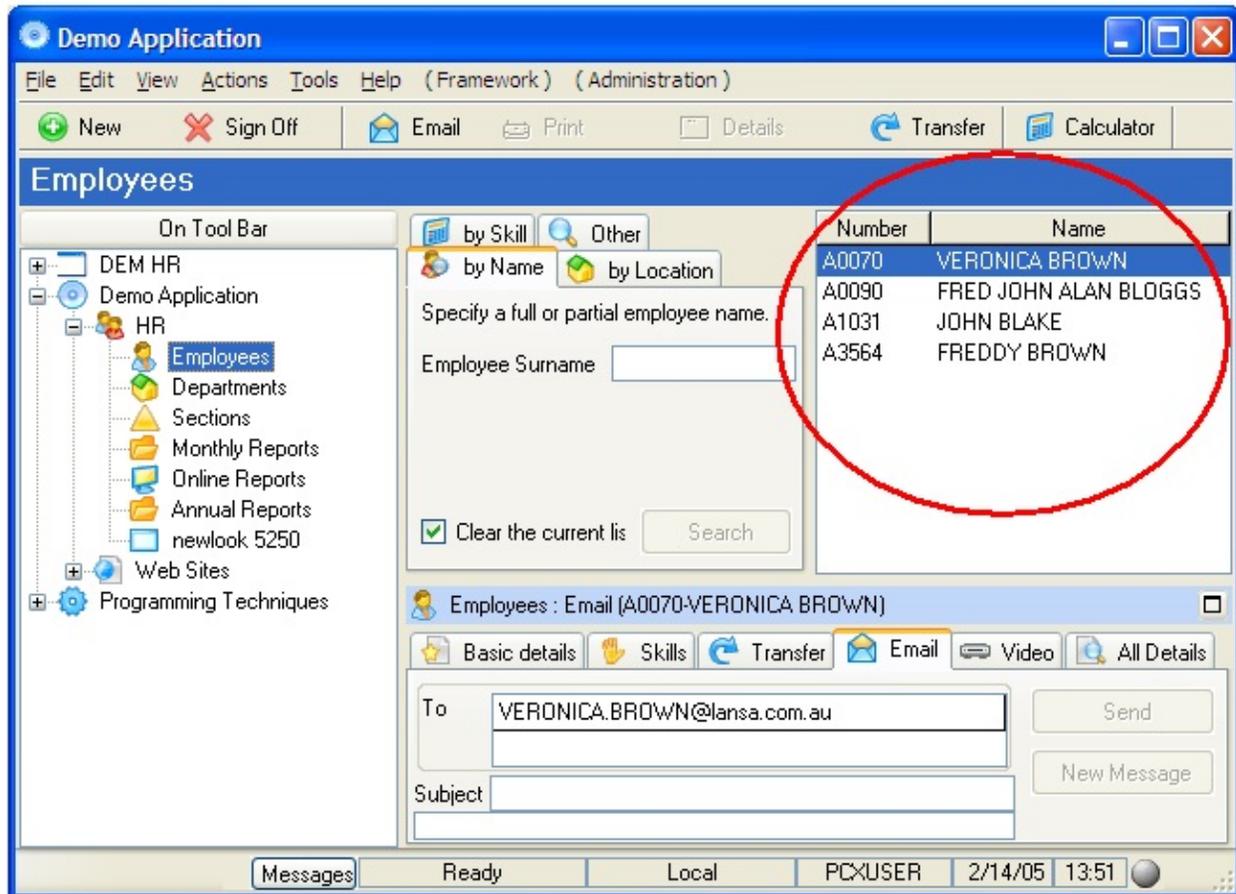


To see examples of using the List Manager and the Framework Manager have a look at the shipped DF_FILTn components.

List Manager and Instance Lists

The component that manages the instance list is called the List Manager.

Each entry in the instance list represents a specific instance of the business object (in this example a specific employee):



See [Basic Instance List Processing](#) for an explanation of how filters, instance lists and command handlers interact.

When a list entry is shown on the screen, it needs to have something that makes it easy for the user to identify it. These visual identifiers often include elements that are very definitely never used as programmatic identifiers such as a name. Most business objects have at least two visual identifiers. See [Visual Identifiers](#).

Each entry in the list must also have something that uniquely identifies it within the list. For example employee number A0070 uniquely identifies employee Veronica Brown. This programmatic identification is required so your filters and command handlers can uniquely identify which instance of a business object they are working with (obviously, name is not necessarily unique). Many

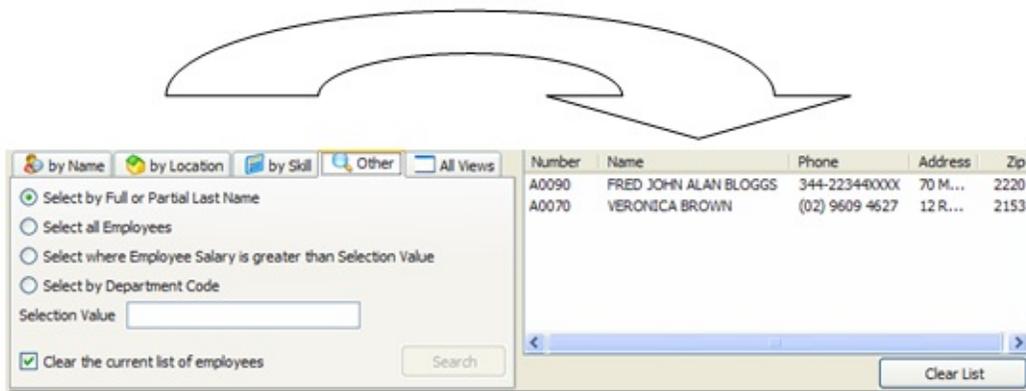
business objects require more than one programmatic identifier to make them unique. See [Programmatic Identifiers](#).

To examine the identifiers used in your Framework application, see [Testing Identifiers](#).

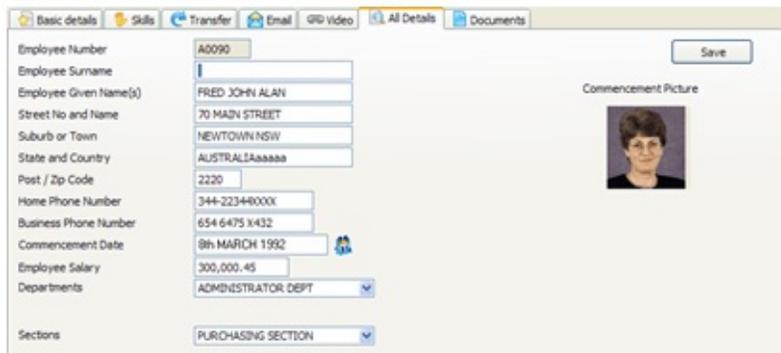
In the Windows Framework the filter can be coded to override the instance list column headings at run time. This can be used to make the instance list suit the result of different filter searches. See [Overriding Instance List Column Headings](#). Also the filter or command handler can be coded to sort the instance list at run time. See [Programmatically Sorting the Instance List](#).

Note that you do not necessarily need to use the standard instance list provided by the Framework. See [Optionally Create Your Own Instance List](#).

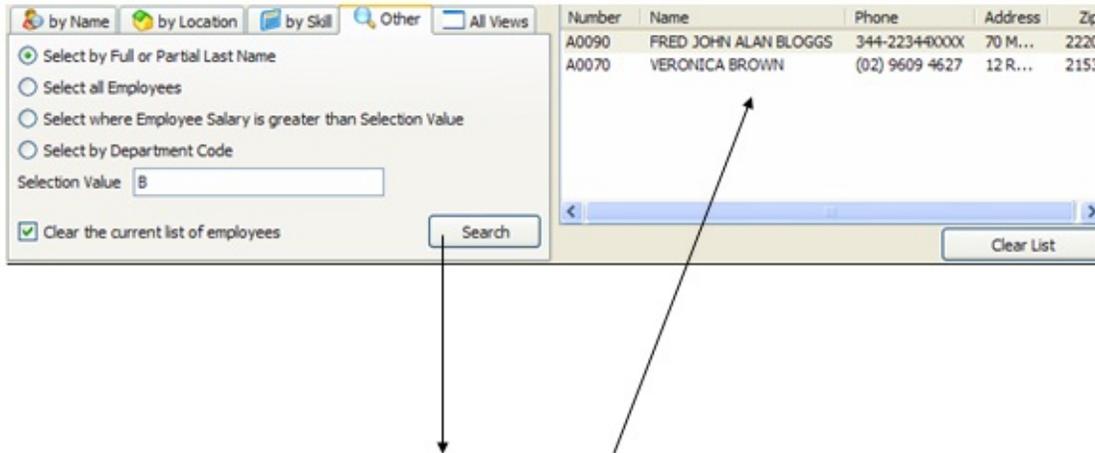
Basic Instance List Processing



In the Framework, filters create lists of business objects such as customers, orders, etc. When the user clicks on a business object instance displayed in the instance list, a set of command handler tabs typically appear, indicating various details about the business object and various actions that users can take on it. For example:



Programmatically this model is quite simple:
Typically a "Search" button in a filter works like this:



* Indicate start of list update and set the Framework to busy

Invoke Method(#avListManager.BeginListUpdate)

* Clear the list first

Invoke Method(#avListManager.ClearList)

* Select the business object(s) that match the search criteria, often from rows
* in a database table and then add them to the instance list

Select Fields() From_File() With_Key() Where()

 Invoke Method(#avListManager.AddtoList)

EndSelect

* Indicate the end of list update

Invoke Method(#avListManager.EndListUpdate)

Structurally, this logic reflects how most filters and instance lists work together.

Visual Identifiers

When you create a filter you will need to decide on the set of visual identifiers. These identifiers allow the user to identify and filter individual business object instances.

When you manage instance lists with the List Manager you use parameters named VisualID1 and VisualID2 to specify (or receive) the visual identifiers assigned to a business object instance.

Both parameters are alphanumeric and have maximum lengths of 32 and 50 characters respectively. You can of course put anything you like into these visual identifiers, including numeric and concatenated information. The [Framework Manager](#) provides a standard method named avMakeAlphaValue to help you convert numeric identifiers such as Product and Customer numbers into alphanumeric form.

Here are some examples of visual identifiers:

Business Object	Chosen VisualID1	Chosen VisualID2
Product	Product Number	Product Description
Customer	Customer Id	Customer Name
Employee	Employee Number	Employee Name (Department - Work Phone Number)
Account	Company Id : Account Number	Account Name
Department	Department Name	Not Required
Section	Department Code – Section Code	Section Name

Remember that visual identifiers are only used so that people can identify things and that they have no bearing whatsoever on how your programs will identify things. You might even dynamically vary the visual identifiers used from object instance to object instance; for example all premium customers get an * in front of their names in VisualID2.

Programmatic Identifiers

You will also need to decide on programmatic identifiers. These identifiers allow your filters and command handlers (i.e.: your programs) to uniquely identify individual business object instances.

In the List Manager you use a set of parameters to indicate your chosen programmatic identifiers:

- AKey1, AKey2, AKey3, AKey4 and AKey5 specify (or receive) the alphanumeric values that identify a business object instance. They are optional and their maximum length is 32 characters (each).
- NKey1, NKey2, NKey3, NKey4 and NKey5 specify (or receive) the numeric values that identify a business object instance. They are optional and their maximum precision is 15,0 (each).

So a business object instance can have up to ten programmatic identifiers, five alphanumeric and five numeric, in any combination. You might just use one or two identifiers, but you can also concatenate several pieces of information into a single AKeyn value, so effectively you can have more keys than 10.

Here are some examples of how you might identify your business objects:

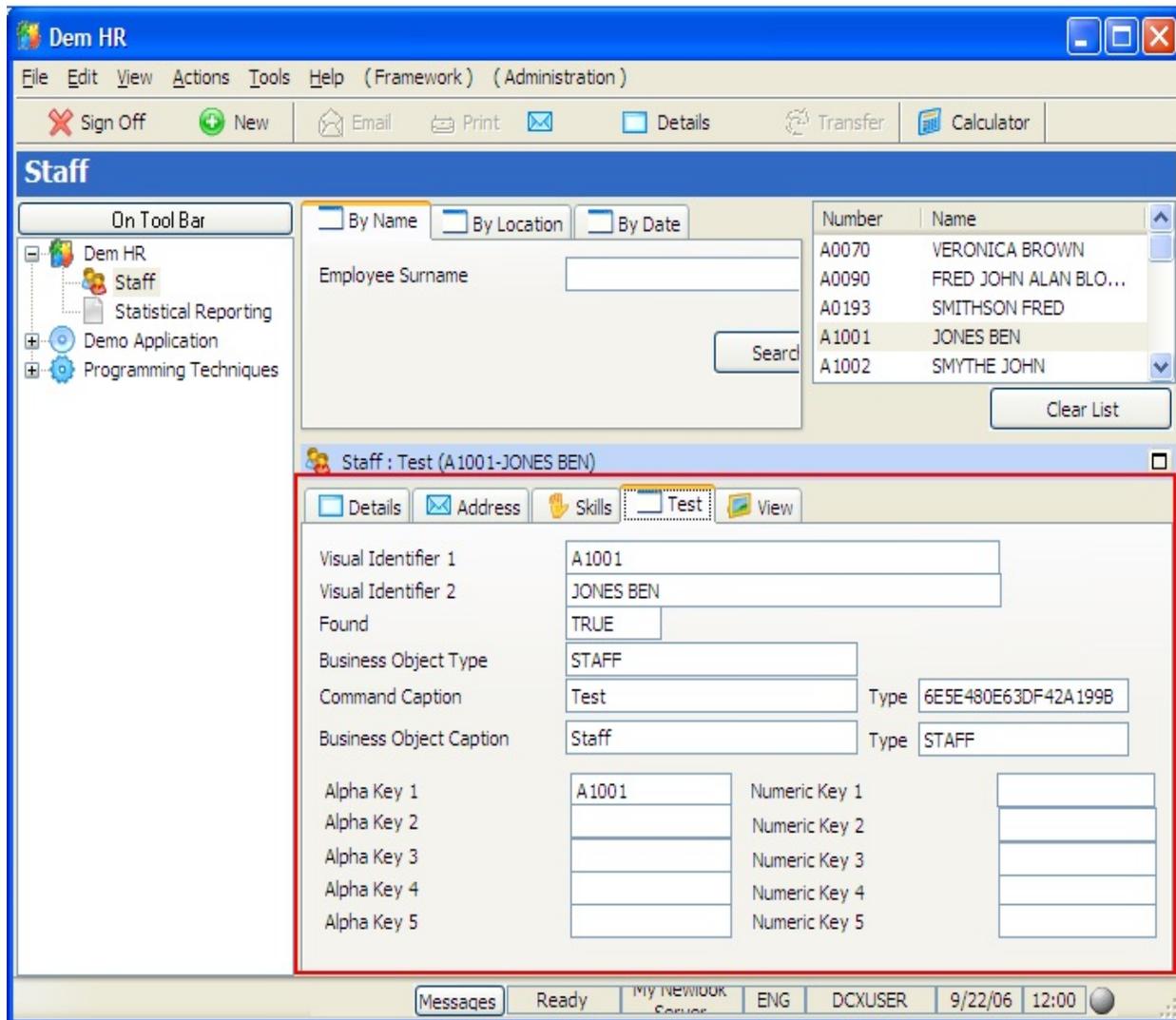
Business Object	AKey1	AKey2	AKey3	AKey4	AKey5	Nkey1	Nkey2
Product						Product Number	
Customer	Customer Number						
Account	Account Number					Company Number	Major Account
Department	Department Code						
Section	Department Code	Section Code					

Remember that these identifiers must uniquely identify an instance of the

business object to your programs but that nobody is going to see them (unless you make them visual identifiers).

Testing Identifiers

You can use a generic VLF command handler to test the visual and programmatic identifiers passed from the instance list. It shows the selected business object and its identifier values:



To use the command handler, copy and paste the sample code to a reusable part, compile it and snap it in as a command handler for any business object:

Function Options(*DIRECT)

```
BEGIN_COM ROLE(*EXTENDS #VF_AC010) HEIGHT(240)
HINT(*MTXTDF_DET1) WIDTH(600)
```

DEFINE_COM CLASS(#VF_ELXVI1.Visual) NAME(#VF_ELXVI1)
DISPLAYPOSITION(1) HEIGHT(19) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(1) TOP(8) USEPICKLIST(False) WIDTH(416)

DEFINE_COM CLASS(#VF_ELXVI2.Visual) NAME(#VF_ELXVI2)
DISPLAYPOSITION(2) HEIGHT(19) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(2) TOP(26) USEPICKLIST(False) WIDTH(417)

DEFINE_COM CLASS(#VF_ELXNK1.Visual) NAME(#VF_ELXNK1)
DISPLAYPOSITION(7) HEIGHT(19) LEFT(272)
PARENT(#COM_OWNER) TABPOSITION(7) TOP(136)
USEPICKLIST(False) WIDTH(247)

DEFINE_COM CLASS(#VF_ELXNK2.Visual) NAME(#VF_ELXNK2)
DISPLAYPOSITION(6) HEIGHT(19) LEFT(273)
PARENT(#COM_OWNER) TABPOSITION(6) TOP(155)
USEPICKLIST(False) WIDTH(247)

DEFINE_COM CLASS(#VF_ELXNK3.Visual) NAME(#VF_ELXNK3)
DISPLAYPOSITION(5) HEIGHT(19) LEFT(273)
PARENT(#COM_OWNER) TABPOSITION(5) TOP(174)
USEPICKLIST(False) WIDTH(247)

DEFINE_COM CLASS(#VF_ELXNK4.Visual) NAME(#VF_ELXNK4)
DISPLAYPOSITION(4) HEIGHT(19) LEFT(273)
PARENT(#COM_OWNER) TABPOSITION(4) TOP(192)
USEPICKLIST(False) WIDTH(247)

DEFINE_COM CLASS(#VF_ELXNK5.Visual) NAME(#VF_ELXNK5)
DISPLAYPOSITION(3) HEIGHT(19) LEFT(273)
PARENT(#COM_OWNER) TABPOSITION(3) TOP(208)
USEPICKLIST(False) WIDTH(247)

DEFINE_COM CLASS(#VF_ELXAK1.Visual) NAME(#VF_ELXAK1)
DISPLAYPOSITION(12) HEIGHT(19) LEFT(11)
PARENT(#COM_OWNER) TABPOSITION(12) TOP(136)
USEPICKLIST(False) WIDTH(249)

DEFINE_COM CLASS(#VF_ELXAK2.Visual) NAME(#VF_ELXAK2)

DISPLAYPOSITION(11) HEIGHT(19) LEFT(11)
PARENT(#COM_OWNER) TABPOSITION(11) TOP(153)
USEPICKLIST(False) WIDTH(249)

DEFINE_COM CLASS(#VF_ELXAK3.Visual) NAME(#VF_ELXAK3)
DISPLAYPOSITION(10) HEIGHT(19) LEFT(11)
PARENT(#COM_OWNER) TABPOSITION(10) TOP(172)
USEPICKLIST(False) WIDTH(249)

DEFINE_COM CLASS(#VF_ELXAK4.Visual) NAME(#VF_ELXAK4)
DISPLAYPOSITION(9) HEIGHT(19) LEFT(11) PARENT(#COM_OWNER)
TABPOSITION(9) TOP(190) USEPICKLIST(False) WIDTH(249)

DEFINE_COM CLASS(#VF_ELXAK5.Visual) NAME(#VF_ELXAK5)
DISPLAYPOSITION(8) HEIGHT(19) LEFT(11) PARENT(#COM_OWNER)
TABPOSITION(8) TOP(208) USEPICKLIST(False) WIDTH(249)

DEFINE_COM CLASS(#VF_ELBOOL.Visual) NAME(#VF_ELBOOL)
CAPTION('Found ') DISPLAYPOSITION(13) HEIGHT(19)
LABELTYPE(Caption) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(13) TOP(44) USEPICKLIST(False) WIDTH(209)

DEFINE_COM CLASS(#VF_ELIDN.Visual) NAME(#VF_ELIDN)
CAPTION('Business Object Type') DISPLAYPOSITION(14) HEIGHT(19)
LABELTYPE(Caption) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(14) TOP(64) USEPICKLIST(False) WIDTH(329)

DEFINE_COM CLASS(#VF_ELXTS.Visual) NAME(#VF_ELXTS)
CAPTION('Type') DISPLAYPOSITION(15) HEIGHT(19)
LABELTYPE(Caption) LEFT(344) MARGINLEFT(30)
PARENT(#COM_OWNER) TABPOSITION(15) TOP(108)
USEPICKLIST(False) WIDTH(159)

DEFINE_COM CLASS(#VF_ELXTL.Visual) NAME(#VF_ELXTL)
CAPTION('Business Object Caption') DISPLAYPOSITION(16) HEIGHT(19)
LABELTYPE(Caption) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(16) TOP(107) USEPICKLIST(False) WIDTH(329)

DEFINE_COM CLASS(#STD_TEXTL.Visual) NAME(#STD_TEXTL)

```
CAPTION('Command Caption') DISPLAYPOSITION(17) HEIGHT(19)
LABELTYPE(Caption) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(17) TOP(84) USEPICKLIST(False) WIDTH(329)
```

```
DEFINE_COM CLASS(#STD_TEXTS.Visual) NAME(#STD_TEXTS)
CAPTION('Type') DISPLAYPOSITION(18) HEIGHT(19)
LABELTYPE(Caption) LEFT(344) MARGINLEFT(30)
PARENT(#COM_OWNER) TABPOSITION(18) TOP(84)
USEPICKLIST(False) WIDTH(159)
```

```
* -----
```

```
* Handle Command Execution
```

```
* -----
```

```
Mthroutine Name(uExecute) Options(*REDEFINE)
```

```
* Do any execution logic defined in the ancestor
```

```
Invoke #Com_Ancessor.uExecute
```

```
* Get the details
```

```
Invoke #avListManager.GetCurrentInstance AKey1(#vf_elxak1)
AKey2(#vf_elxak2) AKey3(#vf_elxak3) AKey4(#vf_elxak4)
AKey5(#vf_elxak5) NKey1(#vf_elxnk1) NKey2(#vf_elxnk2)
NKey3(#vf_elxnk3) NKey4(#vf_elxnk4) NKey5(#vf_elxnk4)
Found(#vf_elBool) VisualId1(#VF_ELXVI1) VisualId2(#VF_ELXVI2)
BusinessObjectType(#vf_elidn)
```

```
* Display other details as well
```

```
Change #Std_texts #com_Owner.avCommandType
```

```
Change #Std_textl #com_Owner.avCommandCaption
```

```
Change #vf_eltxts #com_Owner.avObjectType
```

Change #vf_eltxtl #com_Owner.avObjectCaption

Endroutine

End_Com

Filters and List Manager

Your filter components will control what entries are displayed in the instance list.

The List Manager provides these methods you can use in your filters:

In **Windows** and **WAM** Applications **Description**

Invoke

#avListManager.BeginListUpdate

Is used to indicate to the list manager that you are about to begin updating the instance list. In effect `BeginListUpdate` and `EndListUpdate` are used to define the boundaries of your list update transaction. The list manager will cause the busy light to be displayed when you begin a list update.

Also see [Updating and Deleting Instance List Entries](#).

Invoke #avListManager.ClearList

Is used to clear the current instance list.

Invoke #avListManager.AddtoList

Is used to add a new entry to the instance list or to update an existing one. You need to supply `VisualID1` and `VisualID2` as well as `AKeyn` and `NKeyn` values to this method.

There is also a `Select` parameter available to indicate whether the instance should be selected.

The `AddtoList` method has an optional `BusinessObjectType` parameter which allows filters to add different business object types to the same instance list.

There is also an optional `RowColor` parameter which allows filters to specify the color when adding an

Instance list entry. See [Changing the Color of List Entries \(RowColor\)](#).

Invoke #avListManager.EndListUpdate

Is used to indicate to the list manager that you have completed updating the list. The list manager will turn off the busy light and update the list displayed on the screen.

*Invoke
#avListManager.RemovefromList*

Not currently available for **WAMs**.

Is used to remove an entry from the instance list. AKeyn and NKeyn values are required to uniquely identify which entry is to be removed. If the entry does not exist then the request is ignored.

*Invoke #avListManager.
UpdateListEntryData*

Not currently available for **WAMs**.

Is used to rapidly update the data content of an existing instance list entry. If the entry does not exist then the request is ignored. Does not need to be bracketed by BeginListUpdate and EndListUpdate method calls.

*Invoke #avListManager.
RefreshRelationship*

Not currently available for **WAMs**.

Is used to refresh a child/descendant tree 'cluster' in an instance list that is displayed as a tree. This method causes a tree child node cluster to be refreshed by calling the relationship handler again. Things you need to know about this method:

- If the child node cluster has not been previously expanded, the refresh request is ignored.
- If the child node cluster has been previously expanded, then the relationship handler is called to (re)expand the relationship again. The instance list entries it returns cause existing tree nodes to be

updated, or new entries to be inserted. Tree nodes that existed before the relationship handler was called, but were not returned by it, are deleted from the tree and instance list.

- Typically this method is only used on the lowest child level in a tree.
- If you have trouble using this method, turn on application tracing and review the output it produces.

Also see [Updating and Deleting Instance List Entries](#).

Invoke
#avListManager.AlterColumnHeadings

Is used to override the instance list column headings at run time. This can be used to make the instance list suit the result of different filter searches.

Not currently available for **WAMs**.

Invoke
#avListManager.ResetAlteredHeadings

Is used to remove all overrides the instance list column headings.

Not currently available for **WAMs**.

Invoke #avListManager.DisplaySorted

Is used to sort the instance list by a visible or hidden column, at run time. Up to four sort columns can be specified.

Not currently available for **WAMs**.

Refer to tutorial [VLF013WIN - Signaling Events](#) or [VLF013WAM - Signaling Events](#) for step-by-step instructions of how to update entries in an instance list.

The programming model used to manage instance lists is the same in Windows and Web browser applications, even though the syntax used is different.

This example fills an instance list with Employee details in **Windows** and **WAM** applications:

```
* Start the list updating and clear the current instance list
Invoke Method(#avListManager.BeginListUpdate)
Invoke Method(#avListManager.ClearList)
* Select all employees with specified surname (generically) and add to instance
Select Fields(#SURNAME #GIVENAME #EMPNO) From_File(PSLMST2) \
    Generic(*YES)
Use Builtin(BCONCAT) With_Args(#GiveName #SurName) To_Get(#FullNa
Invoke Method(#avListManager.AddtoList) Visualid1(#Empno) Visualid2(#Fu
    Akey1(#Empno)
EndSelect
* Instance list updating has been completed
Invoke Method(#avListManager.EndListUpdate)
```

Updating and Deleting Instance List Entries

A special parameter Mode(DYNAMIC) can be used on the BeginListUpdate method.

Instance list activities typically fall into two broad modes of processing:

- **Search and Refresh:** Typically a search button clears the instance list and then completely refills it from a user initiated search. This is called Mode(REFRESH) processing because the whole instance list is refreshed. Sometimes the list is not cleared, but progressively added to allow the end-user to easily perform "and" style searches.
- **Individual Entry Updating and Deleting:** Typically a command handler has performed some action that it knows should be reflected into a small number of entries already in the instance list. This is called Mode(DYNAMIC) processing because only a few entries in the instance list need to have their visual content dynamically updated or removed.

Mode(REFRESH) is the default style of list processing.

There's some differences between how Mode(REFRESH) and Mode(DYNAMIC) requests are handled by the Framework:

- When you use the default Mode(REFRESH) method, the visualization of the instance list is performed when the EndListUpdate method is executed. This is done by clearing the entire existing visualization and rebuilding it from scratch. This means selections and focus are lost and trees are collapsed, even if you replace the list with the exactly the same content.
- When you use Mode(REFRESH) you can execute a default command and set instance selections.
- When you use Mode(DYNAMIC) the visualization of the change to the instance list is performed immediately. You can't execute a default command and set or change the instance selection because all you are intending to do is change or remove its visualization, but existing selections and focus are not lost and trees are not collapsed.
- You can't use #ListManager.ClearList when using Mode(DYNAMIC) updating. This will cause an error message to be displayed.

Finally, there's two ways to update an instance list entry: You can use #ListManager.AddtoList or #ListManager.UpdateListEntryData. Both will cause an existing entry to be updated, but only AddtoList will create a new entry when the specified one does not exist.

The main difference between them is:

- UpdateListEntryData only updates the list entry values that you specify as parameters. For example you can just update additional column 3.
- AddtoList updates all the values in the instance list entry using either the values you pass as parameters or their default values. So to update additional column 3 you probably need to pass values for additional columns 1 and 2 as well, and maybe 4, 5 and 6 as well.
- The preceding points mean you should not use UpdateListEntryData in MODE(REFRESH) updates.

Warning:

- Do not attempt to update the identifying keys of an instance list entry either by using UpdateListEntryData or using RemoveFromList followed by an AddtoList. The identifying keys (Akey, NKey) should be values that are not changed by any command handlers. Use additional columns for instance list values that can be changed.

When a Relationship handler is used to dynamically expand the nodes in an instance list displayed as a tree you can use the #avListManager.RefreshRelationship method to programmatically cause a level in the tree to be completely refreshed.

For example, the shipped demonstration filter DF_FILT08 contains this logic listening for the event DEM_EMP_UPDATED

* If an employee update has been triggered

When (= DEM_EMP_UPDATED)

* Get the department and section this employee belonged to at the time they were added to the instance list

Invoke #avListManager.GetCurrentInstance AKey1(#Original_Deptment)
AKey2(#Original_Section) AKey3(#Empno)

* Refresh the tree node that the employee was in originally (this might cause the employee to be removed from the node).

* This method causes the relationship handler function DFREL01 to be called again to refresh the whole tree node.

```
Invoke #avListmanager.RefreshRelationship  
BusinessObjectType(DEM_ORG_SEC_EMP) AKey1(#Original_Deptment)  
Akey2(#Original_Section)
```

* Now see what department and section the employee is in now. If either has changed, update the tree node for the

* the new department/section. If this node has never been expanded this request will be ignored, because the employee

* will be shown later if / when the user decides to expand this node.

```
Fetch Fields(#Deptment #Section) from_file(PslMst) with_key(#Empno)
```

```
If ((#Deptment *ne #Original_Deptment) or (#Section *ne  
#Original_Section))
```

```
Invoke #avListmanager.RefreshRelationship  
BusinessObjectType(DEM_ORG_SEC_EMP) AKey1(#Deptment)  
Akey2(#Section)  
Endif
```

Command Handlers and List Manager

The instance list allows the end-user to select one or more business object instances and then execute a command (i.e. a command handler):

Number	Name
A0070	VERONICA BROWN
A0090	FRED JOHN ALAN BLOGGS
A1031	JOHN BLAKE
A3564	FREDDY BROWN

In your command handler you often need to find out:

- The Current Instance which is the selected instance, if there is only one, or the instance with the focus if there are more.
- The Selected Instances which are all the instances that have been selected.

You can use these List Manager methods in your command handlers:

In Windows and WAM Applications Description

Invoke #avListManager.

GetCurrentInstance

Used to ask the list manager for details of the current business object instance.

If your command handler is designed to handle single business object instances, for example for displaying the details of a specific product, you need to use this method to find out what the current product is.

Invoke #avListManager.

GetSelectedInstance

Used to ask the list manager to return all selected instances in the list. Some command handlers are designed to work with multiple object instances (eg: Printing details of all selected employees).

<p><i>Invoke #avListManager.GetFromList</i> Not currently available for WAMs</p>	<p>Used to ask the list manager to return the unique identifier and the visual identifiers for an instance in the instance list. AKeyn and NKeyn values are required to uniquely identify the instance to be read.</p>
<p><i>Invoke #avListManager.GetInstance</i></p>	<p>Used to ask the list manager to return all instances entries in the list. This method allows command handlers to be designed to work with all object instances returned by a filter.</p>
<p>Invoke #avListManager.AlterColumnHeadings Not currently available for WAMs.</p>	<p>Is used to override the instance list column headings at run time. This can be used to make the instance list suit the result of different filter searches.</p>
<p>Invoke #avListManager.ResetAlteredHeadings Not currently available for WAMs.</p>	<p>Is used to remove all overrides the instance list column headings.</p>
<p>Invoke #avListManager.DisplaySorted Not currently available for WAMs.</p>	<p>Is used to sort the instance list by a visible or hidden column, at run time. Up to four sort columns can be specified.</p>

In **Windows** applications the list manager also signals a ListSelectionChanged event which you might want to listen to use in more complex command handlers. It is signaled whenever the selected set of object instances changes. In Web browser applications the same event is handled by completely (re)executing your command handler. There are examples of this type of event handling in the shipped demonstration and programming techniques

applications.

Note: You should not attempt to change the content or selection in the instance list in any way within a ListSelectionChanged event handler.

The programming model used to manage instance lists is the same in Windows and Web browser applications. This is how you would find out what the current Employee is an Employees instance list in a **Windows** or **WAM** application:

```
Invoke #avListManager.GetCurrentInstance AKey1(#EMPNO)
```

This is how you would find all the Employees that are currently in an instance list whether they are selected or not:

```
Def_cond *RetOkay '#RetCode = OK'  
Invoke #avListManager.GetInstance First(TRUE) Akey1(#Empno)  
ReturnCode(#RetCode)  
DoWhile *RetOkay  
Fetch Fields(.....whatever ....) From_File(PSLMST) With_Key(#Empno)  
Invoke #avListManager.GetInstance First(FALSE) Akey1(#Empno)  
ReturnCode(#RetCode)  
EndWhile
```

Current/Selected Instance

The shipped sample command handler DF_DET14 shows you the current and the selected instances in any instance list.

In the following example DF_DET14 has been assigned as the command handler for the Status command. It shows the current instance and all the selected instances in an Employee instance list:

The screenshot shows a software interface with a search panel on the left and a main display area on the right. The search panel has tabs for 'by Skill', 'Other', 'by Name', and 'by Location'. A text input field labeled 'Employee Surname' contains the letter 'S'. Below it is a 'Search' button and a checked 'Clear the current list' checkbox. The main display area shows a table of employees with columns 'Numbe' and 'Name'. The table contains the following data:

Numbe	Name
A0193	FRED SMITHSON
A1002	JOHN SMYTHE
A1003	Robert SMITHE
A1004	PAUL SMITHSON
A1005	PETER SMITHS
A1006	JACK SMITHERS
A1007	GEORGE SNELL
A1008	ALLAN SNEDDON
A1009	DAMIAN SNASHALL

The row for A1004 - PAUL SMITHSON is highlighted with a blue background and a dotted border, indicating it is the current instance. Below the table, the interface shows a status command handler for 'Employees : Status (A1004-PAUL SMITHSON)'. It has tabs for 'All Details', 'Status', 'Basic details', 'Skills', 'Transfer', 'Email', and 'Video'. The 'Status' tab is active, displaying the text: 'The CURRENT Business Object Instance is A1004 - PAUL SMITHSON' and 'The currently SELECTED Business Object Instances are ...'. Below this text is a table with two columns: 'Visual Identifier 1' and 'Visual Identifier 2'.

Visual Identifier 1	Visual Identifier 2
A1005	PETER SMITHS
A1006	JACK SMITHERS
A1004	PAUL SMITHSON

Note the dotted lines around Paul Smithson indicating it is the current instance. Try assigning DF_DET14 as the command handler for a business object to see how GetCurrentInstance and GetSelectedInstance works. Remember that the command you assign it to has to be an [Instance Command](#).

Authority to Instances

GetCurrentInstance and GetSelectedInstance will return the next instance (regardless of whether or not the current user is authorized to the instance in the current command context). Preventing access to the instance at this level is an application issue, which can may be handled in at least two different ways:

- Modify the filter to invoke #Com_Owner.avCheckInstanceAuth and based on the returned values for the instance, decide whether to add the instance to the instance list.
- Modify the command handler to invoke #Com_Owner.avCheckInstanceAuth and based on the returned values for the instance and the command, decide whether some or all fields on the command handler should become invisible or read-only for the current user.

More about Instance Lists

Please note that:

- In Windows applications instance Lists are saved when the Framework is closed and initialized (depending up system settings) when it is restarted. They are saved on a user basis.
- In Web browser applications instance list are neither saved nor initialized.
- Only business objects have instance lists. Command handlers designed to work at the Application or Framework level (i.e. not with business objects) must not reference instance lists. In Windows command handlers, you need to avoid any references to object #avListManager as it will not exist.

Adding Additional Columns to Instance Lists

[VLF009WAM - Adding Instance List Columns in WAM Applications](#) [VLF009WIN - Adding Instance List Columns in Windows Applications](#)

Sometimes you might want to add additional columns to an instance list:

Name	Address Line 1	Address Line 2	Address Line 3	Zip Code	Business Phone	Department	Section	Start Date
BROWN,VERONICA	12 Railway Street	Baulkham Hills	NSW Australia	2153	(02) 9647 2788	INF	DV	28th January 1990
BLOGGS,FRED JOHN ALAN	70 MAIN STREET	NEWTOWN NSW	AUSTRALIA	2220	654 6475 X432	FLT	03	3rd August 1992
JONES,BEN	144 Frog	PYMBLE.	NSW.	2001	798 0543	ADM	01	1st February 1988
SMITHE,Robert	29 Arthur Road,	DEE WHY.	NSW.	2000	406 6395	FLT	02	21st December 1985
WOODS,BRADLEY	59 Darley Road,	BEXLEY.	NSW.	2030	789 4562	ADM	01	12th December 1984
ROBINSON,MARY	14 Whitby Road,	ST IVES.	NSW.	2005	456 1852	ADM	01	1st May 1986
BLAKE,JOHN	3 Woodbury Road	Winston Hills	NSW Australia	2100	(02) 9922 5588	MIS	EI	18th May 1996
MRS BRICK,GILL	22 Moton Street	Marrickville	NSW	2090	324 444	ADM	01	1st May 1994
REDFORD,ROBERT	122 Arthur Street	North Sydney	NSW Australia	2060	9573188	ADM	01	19th February 1995
BROWN,FREDDY	121 SMITH STREET	Newtown		2153	(02) 456-5678	ADM	04	31st May 1995

Displaying Additional Columns

For a Windows application:

- You can display additional columns in the shipped instance list. You specify the columns on the Instance List/Relations tab of the properties folder of the business object you are working with.

or

- Another option is to replace the standard instance list with your own instance list reusable part. There is a Program Coding Assistant that will generate a complete instance list.
- To attach your own instance list display the Instance List/Relations tab sheet and specify the name of the reusable part you have created in the Snap in Instance List Browser field.

For a **Web** browser application:

- You can display additional columns in the shipped instance list. You specify the columns on the Instance List/Relations tab of the properties folder of the business object you are working with.

If you look at the properties folder for the shipped demonstration business objects you will see the layouts of the instance list defined like this:

Sequence	Type	Caption	Width % (Total 100%)	Decimals	Edit Code	Date/Time Output Format	UTC Conversion
10	VISUALID1	Name	20		Default	SY5FMT8	Local -> Local
20	VISUALID2	Code / Id	10		Default	SY5FMT8	Local -> Local
30	ACOLUMN1	Address Line 1	10		Default	SY5FMT8	Local -> Local
40	ACOLUMN2	Address Line 2	10		Default	SY5FMT8	Local -> Local
50	ACOLUMN3	Address Line 3	10		Default	SY5FMT8	Local -> Local
70	NCOLUMN1	Zip Code	10		Default	SY5FMT8	Local -> Local
75	ACOLUMN4	Business Phone	10		Default	SY5FMT8	Local -> Local
80	ACOLUMN5	Home Phone	10		Default	SY5FMT8	Local -> Local

Programming your instance list

Date/Time Additional Column Programming Example

Programming your instance list

The logic you use to work with your instance list is the same in Windows or Web browser filters even though the syntax is different.

The shipped **Windows** Sections filter DF_FILT4 uses this code to add details to the instance list:

```
Select Fields(#DEPARTMENT #SECTION #SECDESC #SECPHBUS #SECPC  
Invoke Method(#avListManager.AddtoList) Akey1(#Deptment) Akey2(#Secti  
    VisualID1(#Section) VisualID2(#SecDesc) AColumn1(#SecDesc)  
    AColumn2(#SecPhBus) NColumn1(#SecPCode)  
Endselect
```

The shipped **WAM** filter DM_FILT4 uses functionally identical code structured like this:

```
Select Fields(#DEPARTMENT #SECTION #SECDESC #SECPHBUS #SECPC  
#avListManager.AddtoList Visualid1(#DEPARTMENT) Visualid2(#SECTION) A  
    AColumn1(#SecDesc) AColumn2(#SecPhBus) NColumn1(#SecPCode)  
Endselect
```

In both cases the additional columns are added to the instance list by reference to properties named like AColumn, NColumn and DColumn.

AColumn is used to identify alphanumeric values and has a maximum length of 100 characters. NColumn is used to identify numeric values and has a maximum precision of 30,9. DColumn is used to identify date or date/time values and are input as alphanumeric values with a maximum length of 19 characters. Date or date/time values passed to the DColumn property must also be in ISO format, i.e., CCYY-MM-DD or CCYY-MM-DD HH:MM:SS. However different display formats can be selected on the Instance List/Relations tab.

Using the standard shipped instance list, in both Windows applications and Web browser applications, you can have at most 10 alphanumeric columns, 10 numeric columns and 5 date or date/time columns.

The number of additional columns you have will affect the performance and storage space used by your application, so use additional columns with care.

Also see [Instance List with more than 10 alphanumeric and/or 10 numeric additional columns](#).

[Date/Time Additional Column Programming Example](#)

Date/Time Additional Column Programming Example

Because Date or Date/Time values passed to DColumn properties must be in ISO format, values from LANSA fields need to be formatted before they are passed. Below is an example of adding Date values to the instance list from an RDML VLF filter.

```
Select Fields(#EMPNO #GIVENAME #SURNAME #DEPARTMENT
#SECTION #ADDRESS1 #PHONEHME #POSTCODE #STARTDTE)
From_File(PSLMST2) With_Key(#SURNAME) Generic(*YES)
Use BConcat (#GiveName #SurName) #FullName
EXECUTE Subroutine(DateToISO) With_Parms(#STARTDTE #VF_ELDTS)
Invoke Method(#avListManager.AddtoList) Visualid1(#EMPNO)
Visualid2(#FullName) Akey1(#DEPARTMENT) Akey2(#SECTION)
Akey3(#EMPNO) AColumn1(#ADDRESS1) AColumn2(#PHONEHME)
NColumn1(#POSTCODE) DColumn1(#VF_ELDTS)
Endselect
```

```
SUBROUTINE Name(DateToISO) Parms((#RETDAT *RECEIVED)
(#VF_ELDTS *RETURNED))
DEFINE Field(#D_Char8) Type(*CHAR) Length(8)
DEFINE Field(#D_Num8) Reffld(#DATE8)
DEFINE Field(#D_Year) Type(*CHAR) Length(4)
DEFINE Field(#D_Month) Type(*CHAR) Length(2)
DEFINE Field(#D_Day) Type(*CHAR) Length(2)
OVERRIDE Field(#D_Year) To_Overlay(#D_Char8)
OVERRIDE Field(#D_Month) To_Overlay(#D_Char8 5)
OVERRIDE Field(#D_Day) To_Overlay(#D_Char8 7)
USE Builtin(CONVERTDATE_NUMERIC) With_Args(#RETDAT B J)
To_Get(#D_Num8)
USE Builtin(NUMERIC_STRING) With_Args(#D_Num8 N)
To_Get(#D_Char8)
USE Builtin(CONCAT) With_Args(#D_Year '-' #D_Month '-' #D_Day)
To_Get(#VF_ELDTS)
ENDROUTINE
```

RDMLX Examples

In RDMLX filters the processing is much easier because intrinsic functions can be used, with the following as an example of what is required to convert RDML field #STARTDTE.

```
* Convert #STARTDTE to the correct format for Date/Time columns
```

```
EXECUTE Subroutine(DateToISO) With_Parms(#STARTDTE 'DDMMYY'  
#VF_ELDTS)
```

```
* Format conversion subroutine definition
```

```
SUBROUTINE Name(DateToISO) Parms((#RETDAT *RECEIVED)  
(#DATEFMT *RECEIVED) (#DATEOUT *RETURNED))  
DEFINE Field(#DATEFMT) Reffld(#VF_ELDTFM)  
DEFINE Field(#DATEOUT) Reffld(#VF_ELDTS)  
If (#RETDAT.isdate(#DATEFMT))  
#DATEOUT := #RETDAT.AsDate(#DATEFMT).AsDisplayString(ISO)  
ELSE  
MESSAGE Msgtxt('Date is not in the correct format')  
Endif  
ENDROUTINE
```

And converting RDMLX field in RDMLX filters is even easier with only the following required to convert Date or DateTime fields to the correct format. In this case the RDMLX Date field #VF_ELDAT.

```
#VF_ELDTS := #VF_ELDAT.AsDisplayString(ISO)
```

Overriding Instance List Column Headings

In the Windows Framework the filter can override the instance list column headings, using the `avListManager.AlterColumnHeadings` method. For example:

```
invoke #avListManager.AlterColumnHeadings ForVisualID1('My Visual ID 1  
Column Heading') ForVisualID2('My Visual ID 2 Column Heading')  
ForAColumn1('My Additional Alpha Column  
1 Heading') ForNColumn1('My Additional Numeric Column 1 Heading')  
ForBusinessObject(<<The user object name/type of the business object>>)
```

The overrides are cumulative.

If the instance list contains a tree of business objects, an override to the column headings for any business object in the tree can be specified. However, the overrides apply only to the business object in the current tree, not to the business object elsewhere in the Framework.

If you want to remove all overrides and revert all the column headings to what they originally were, you can use `avListManager.ResetAlteredHeadings`:

```
invoke #avListManager.ResetAlteredHeadings
```

This removes all overrides for any business object in the current tree.

Programmatically Sorting the Instance List

In the Windows Framework the filter or command handler can be coded to sort the instance list using the `avListManager.DisplaySorted` method:

```
invoke #avListManager.DisplaySorted ForBusinessObject(EMPLOYEES)  
ByColumn_1(ACOLUMN1) Ascending_1(FALSE)
```

```
invoke #avListManager.DisplaySorted ForBusinessObject(EMPLOYEES)  
ByColumn_1(ACOLUMN1) ByColumn_2(AKEY2)
```

The `ByColumn` parameter can be any of these values:

VISID1

VISID2

AKEY1 to AKEY5

NKEY1 to NKEY5

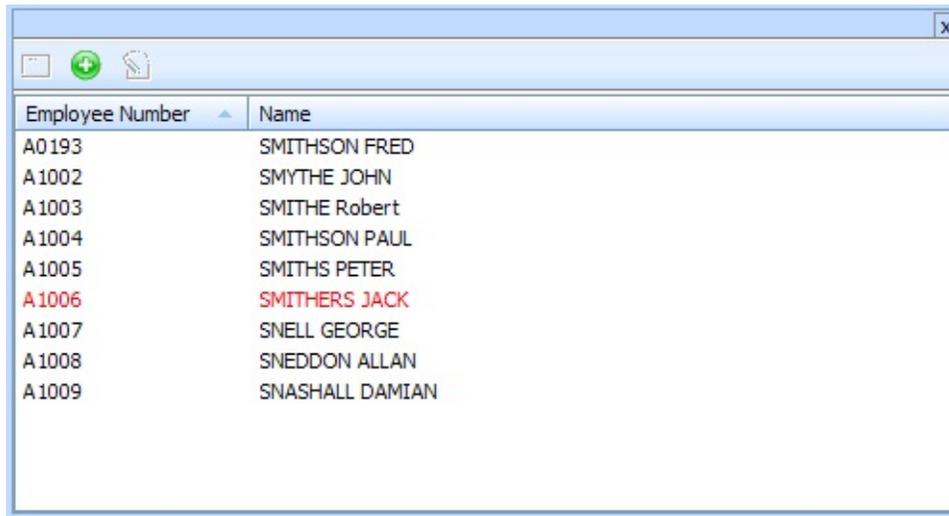
ACOLUMN1 to ACOLUMN10

NCOLUMN1 to NCOLUMN10

DCOLUMN1 to DCOLUMN5

Changing the Color of List Entries (RowColor)

The color of a list entry can be changed in all environments. It can be changed from a Filter, Command Handler, Relationship Handler, and within a Snap In instance list, if coded for.



The screenshot shows a window with a table of employee data. The table has two columns: 'Employee Number' and 'Name'. The row for 'A1006 SMITHERS JACK' is highlighted in red, while the other rows are in a light yellow background. The window title bar includes a close button (X) and standard navigation icons.

Employee Number	Name
A0193	SMITHSON FRED
A1002	SMYTHE JOHN
A1003	SMITHE Robert
A1004	SMITHSON PAUL
A1005	SMITHS PETER
A1006	SMITHERS JACK
A1007	SNELL GEORGE
A1008	SNEDDON ALLAN
A1009	SNASHALL DAMIAN

In VLF.WIN environment

In VLF.WEB environment

In VLF.WIN environment

#avListManager.AddToList and UpdateListEntryData have a parameter, called RowColor which receives a string that must be:

- 1) blank / unspecified - leave the color as it is.
- 2) "DEFAULT" - set the color back to what it would normally be.
- 3) the name of a visual style that has been defined in the IDE, and that has been enrolled in the VLF (in your equivalent of UF_SYSTM).

To use this feature in Windows, you must first create a visual style for each color you want to use. You can copy from the base visual style used by your framework. Edit the visual style's VALUE and set the normbackcolor to what you want:

```
Function Options(*DIRECT)
begin_com role(*EXTENDS #PRIM_VS) default(#SCHEME)
define_com class(#PRIM_VSS) name(#SCHEME) captions(#CAPTION)
titles(#CAPTION) values(#VALUE)
define_com class(#PRIM_VSI) name(#CAPTION) facename('VL Shell')
fontsize(8) normbackcolor(192:215:249) textcolor(MenuText)
define_com class(#PRIM_VSI) name(#VALUE)
alternbackcolor(192:215:249) bordercolor(WindowText) borderstyle(3DLeft)
errorbackcolor(192:215:249) facename('VL Shell') fontsize(8)
normbackcolor(Red) textcolor(MenuText)
End_Com
```

(You could also change other properties of the VALUE if necessary)

Save the visual style and enrol it in the VLF by adding a line to your version of UF_SYSTM like this:

```
*
=====
* This IIP method (avEnrollVisualStyles) enroll all framework user
* visual styles. To use this routine define the visual style into the LANSA
* repository in the usual manner and then add a new invocation of
* avEnrollVisualStyle to the following routine. If the visual style is
* subsequently changed only this component needs to be recompiled to
* effect the change into all other components.
```

```

Mthroutine Name(avEnrollVisualStyles) Options(*REDEFINE)
...
* Test Style with red background for highlighting instance list rows
Invoke Method(#Com_Owner.avEnrollVisualStyle) Style(#<<name of my red
visual style>>) Caption('Base Style with red')

Endroutine

```

Compile your version of UF_SYSTEM.

Now you can specify a row color when adding or updating instance list entries from a filter or command handler, like this:

```

Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#DEPARTMENT) AKey2(#SECTION)
Akey3(#Empno) RowColor(<<name of my red visual style>>)

```

If you subsequently want to change that row back to its original color do this:

```

Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#DEPARTMENT) AKey2(#SECTION)
Akey3(#Empno) RowColor(DEFAULT)

```

If you want to leave the color unchanged do not specify the RowColor:

```

Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#DEPARTMENT) AKey2(#SECTION)
Akey3(#Empno)

```

or set it to *blanks:

```

Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#DEPARTMENT) AKey2(#SECTION)
Akey3(#Empno) RowColor(*blanks)

```

[Relationship Handler](#)

[Snap in Instance List](#)

Relationship Handler

You can also set the row color from a relationship handler, using a line like this:

```
EXECUTE SUBROUTINE(SETROWCOL) WITH_PARMs(<<name of my  
red visual style>> "color:RED")
```

(the second parameter, "color:RED", is for WEB environments)

Snap in Instance List

If you have coded a snap in instance list handler, it can be modified to handle changes to row color like this:

```
define_com #prim_vs #TempVisualStyle Reference(*dynamic)
...
* -----
* Redefine the standard uAddListEntry method
* -----
Mthroutine Name(uAddListEntry) Options(*Redefine)
...
* Add the details to the instance list

Add_Entry #Inst_List

Set_Ref #KeyedGridItems<#uInstanceIdentifier> To(#Inst_list.CurrentItem)

* Set the visual Style based on the RowColor
invoke #avListManager.uGetRowColor uUII(#uInstanceIdentifier)
RowColorStyle(#TempVisualStyle)
if_ref #TempVisualStyle is_not(*null)

* the filter has set a RowColor to a Visual Style, or has set it back to
DEFAULT
set #Inst_List.Currentitem VisualStyle(#TempVisualStyle)

else
* The filter has not specified the RowColor or has specified it as blank - no
change to row style - do nothing

endif

endroutine
```

You can handle changes to row color similarly in mthroutine UpdateListEntryData.

In VLF.WEB environment

The RowColor parameter is a CSS string as specified by the CSS standards. No visual style needs to be created or enrolled.

The RowColor parameter receives a string that must be:

- 1) blank / unspecified - leave the color as it is.
- 2) "DEFAULT" - set the color back to what it would normally be.
- 3) a CSS string, for example: "color:RED;background-color:BLUE;font-style:italic "

The color name can be: AQUA, BLACK, BLUE, FUCHSIA, GRAY, GREEN, LIME, MAROON, NAVY, OLIVE, PURPLE, RED, SILVER, TEAL, WHITE, YELLOW or #nnnnnn (a Hex color name - e.g. #FF0000)

Note that in the .Net framework only these CSS attributes are supported:

- color
- background-color
- font-style
- font-size
- font-weight
- font-family

Shorthand CSS format is also supported, for example "font:italic bold 12px".

[WAMs](#)

[Relationship Handler](#)

WAMs

The AddToList and UpdateListEntryData mthrouines have a parameter, called RowColor, which is used to control the color of rows. This is how you use it:

```
Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)  
Visualid2(#UF_VisID2) AKey1(#DEPARTMENT) AKey2(#SECTION)  
Akey3(#Empno) RowColor(color:RED;background-color:BLUE)  
Select(FALSE)
```

Relationship Handler

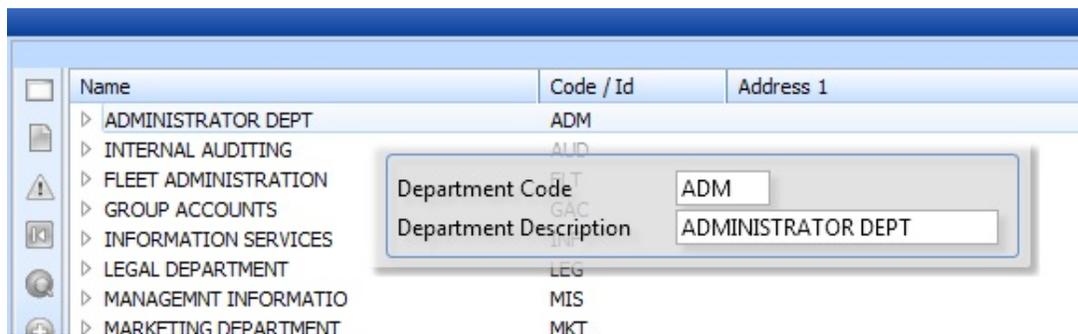
You can also set the web row color from a relationship handler using a statement like this:

```
EXECUTE SUBROUTINE(SETROWCOL) WITH_PARM('XXXXXX'  
"color:RED;background-color:BLUE")
```

(The first parameter, "XXXXXX", is for VLF.WIN environments.)

Popup Panel Hints in the Instance List

A popup panel associated with an instance list entry can be made to appear when the user hovers over an instance list entry. This feature is only available when the Framework is run in Direct-X mode.



In the VLF there is a business object property ([Popup Panel Name](#)) on the instance list tab which stores the name of a reusable part which is the panel inside a popup panel.

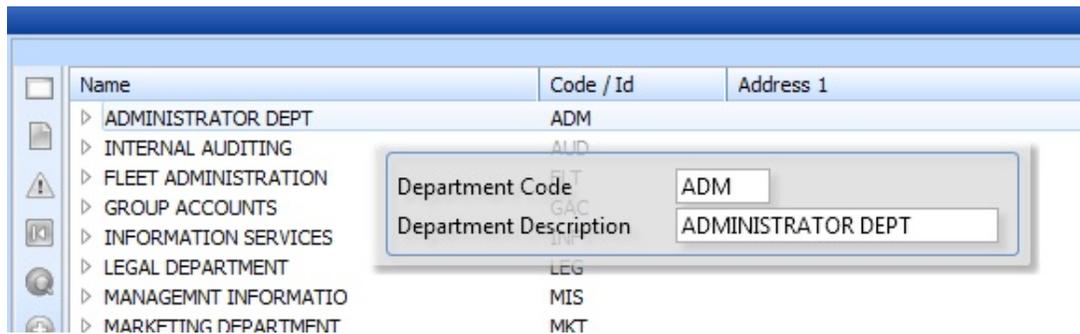
If this property is blank (the default), it indicates that there will be no popup panel for entries of this business object type (via any instance list).

There is also a property of the business object ([Enable Popup Panels](#)) on the instance list tab that can be used to enable or disable popups for any business object within the tree (default is false). If this setting is true, any business object that has an enabled popup panel will display it when the user hovers over an entry of that business object type.

The end-user can disable or enable popups globally, by right-mouse clicking on the instance list and using the context menu. The property will be remembered for the user from session to session.

[How to Use the Instance List Popup Feature](#)

How to Use the Instance List Popup Feature

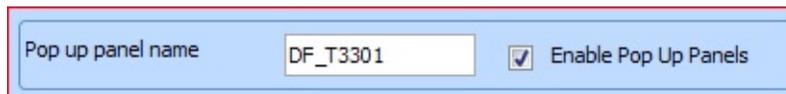


See shipped examples DF_T3301 DF_T3302 DF_T3303.

Create a reusable part with an ancestor of VF_AC021. It is probably easiest to copy the source code of a simple popup panel like DF_T3301, and modify it.

Compile your reusable part.

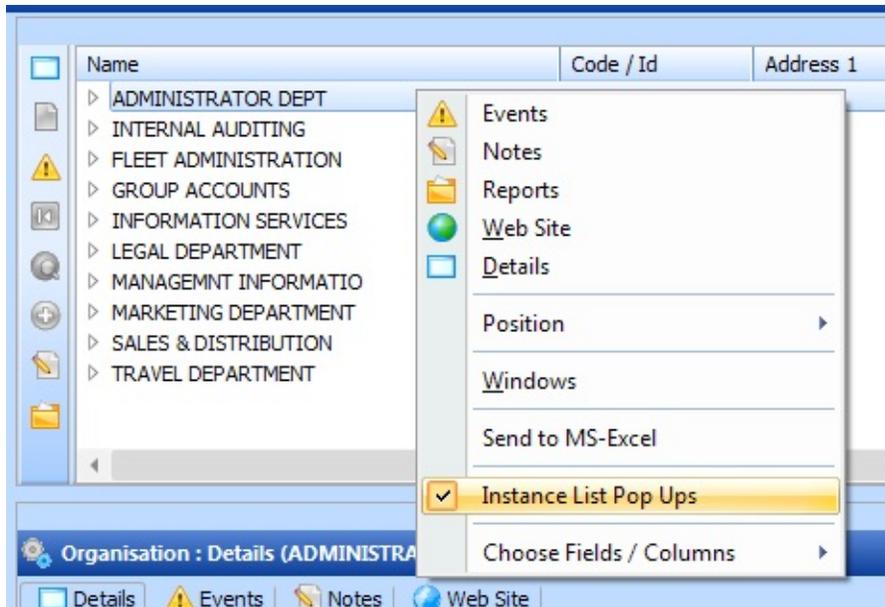
Now snap it in to a business object by opening the business object's properties, and going to the instance list tab, and setting the properties shown below.



Save the framework and run it

If you run the framework in Direct-X mode, and mouse over the instance list over an item of the correct business object type, you should see the popup.

The user can enable or disable all available popups by right mouse clicking instance list popups, as shown.



In the popup panel the mandatory uLoad routine has the following parameters:

Akey1-5 *input

NKey1-5 *input

Visual ID 1

Visual ID 2

User Object Type of the business object *input

subtype *input

EdgeLeft *input

EdgeRight *input

Left *both

Top *both

The first 6 properties are the standard framework identifiers for instance list entries.

The EdgeLeft / EdgeRight properties are the position of the left and right edges of the instance list. These values can be used to set the popup's Left position relative to the instance list, by returning the Left parameter. (See DF_T3303)

* you could ignore the cursor position and locate the popup relative to the list's left or right boundaries

#Left := (#EdgeRight - 200)

Left and Top allow the panel designer to re-position the popup based on the cursor position. The values received are based on the cursor's current location. The designer's logic can either increment these, or replace them with a position that ignores the cursor position. (See DF_T3301)

* If you want to offset the popup from the cursor, increment these values

#Left += 50

#Top += 10

For now this feature is limited to the primary instance list, non-cluster entries.

Instance Lists with Different Types of Objects

You can create instance lists that contain more than one type of object. You do this by defining relationships between business objects. The relationships can either be peer-to-peer or parent-child.

For step-by-step instructions of how to create parent-child relationships refer to one of these tutorials: [VLF011WIN - Creating a Parent Child Instance List](#) or [VLF011WAM - Creating a Parent Child Instance List](#). You may also want to have a look at the Advanced Instance List examples in the Advanced section of the Programming Techniques application.

The Advanced Instance Lists (1) example shows the instance list presented as a tree involving different types of business objects: products and two child objects Orders and Suppliers. Orders also have the child object Shipments:

The screenshot shows the 'Advanced Instance Lists (1)' application window. The window title is 'Advanced Instance Lists (1)'. The menu bar includes 'File', 'Edit', 'View', 'Help', and 'Windows (Framework) (Administration)'. The toolbar contains icons for 'New', 'Reports', 'Web Site', 'About', 'Address', 'Resources', 'Organization', 'Bookings', and 'Charges'. A 'Quick Find ...' search box is located on the right side of the toolbar.

The main content area is titled 'Advanced Instance Lists (1)'. It features a tree view on the left side under 'Favorites' > 'HR Demo Application' > 'Programming Techniques' > 'Advanced'. The tree view shows a hierarchy of objects: 'Product' (PRODUCT0001), 'Orders' (ORDER0001, ORDER0002), 'Shipments' (SHIPMENT0001, SHIPMENT0002), and 'Suppliers'. The 'Orders' and 'Shipments' objects are highlighted with a red box, indicating their parent-child relationship.

In the center, there is a text box titled 'About this example' with the following text: 'Until now you have probably only cre object (eg: instance lists of Products, instance lists may be set up to contain more than one type of object. This example uses an instance list that contains more than one type of object. They are named PRODUCT, ORDER, and SUPPLIER. They are also related: -> a PRODUCT may have child SUPPLIER -> a PRODUCT may have child ORDER'. Below this text is a 'Create Instance List' section with a 'Limit to' field set to '10' and a 'business objects' button.

The status bar at the bottom shows 'Messages', 'Ready', 'Local', 'ENG', 'VLFPGMLIB', '21/08/11', and '17:05'.

Parent-Child Relationships

Peer-to-Peer Relationships

Defining the Object Relationships

Reviewing the Instance List Relationships Holistically

Prototyping and Displaying Instance Lists

Adding Entries to the Instance List

Hiding Objects in Navigation Pane

Work with Hidden Child Objects

Changing the order of child business objects in the instance list tree

Things to Watch Out For

Application Tracing for Relationship Handlers

Sample Relationship Handler Function

Parent-Child Relationships

The shipped Demonstration application supplies a parent-child relationship example in the Organizations business object.

In this example the Organizations business object instance list is defined so that it may contain different types of business objects:

DEPARTMENTS

SECTIONS

EMPLOYEES (Resources)

The screenshot shows the 'Organizations' application window. The left pane displays a tree view with the following structure:

- Favorites
 - HR Demo Application
 - Organizations
 - Resources
 - Programming Techniques
 - Basic
 - Advanced
 - Fast Parts
 - Prompting
 - Administration
 - Users and Authorities

The main pane displays a table with the following data:

Name	Code / Id	Address 1	Ad...	Adresse
ADMINISTRATOR DEPT	ADM			
INTERNAL AUDITING	AUD			
FLEET ADMINISTRATION	FLT			
Sections				
ACCOUNTING	03	21 Redmyre Road,	ST...	NSW.
PURCHASING	02	25 Lovell Road,	EA...	NSW.
ADMINISTRATION	01	81 Parklands Ro...	N...	NSW.
Resources				
TURNER, JACK	A1016	22 Wentworth A...	W...	NSW.
GROUP ACCOUNTS	GAC			
INFORMATION SERVICES	INF			
LEGAL DEPARTMENT	LEG			

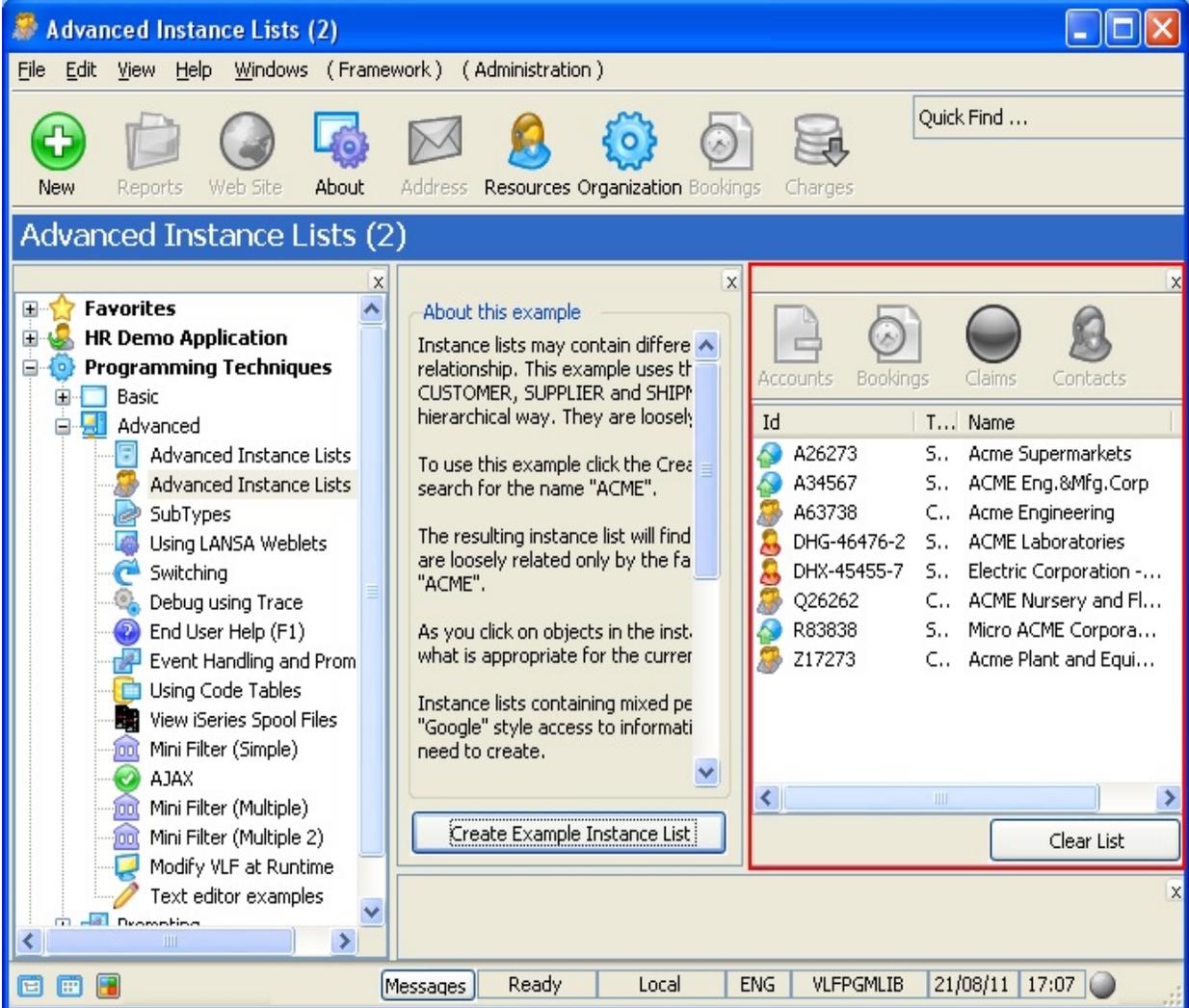
The 'Section : Details (ADMINISTRATION-01)' pane is active, showing a 'Details' tab and a 'Choose Statistic to Graph' dropdown menu set to 'Salary Levels Bar Graph'. A 'Refresh' button is located at the bottom right of this pane.

The status bar at the bottom of the application shows: Messages | Ready | Local | ENG | DCXUSER | 9/21/07 | 13:43

Peer-to-Peer Relationships

Peer objects are used when a single instance list can contain different types of objects that have no hierarchical relationship. In this way you can start building filters that act like Google over corporate data (ie: you just search for a string and a list of the different object types that match is produced).

For an example of an instance list with peer relationships see the Advanced Instance Lists(2) sample in Advanced Programming Techniques:



The screenshot shows the 'Advanced Instance Lists (2)' application window. The window title is 'Advanced Instance Lists (2)'. The menu bar includes 'File', 'Edit', 'View', 'Help', and 'Windows (Framework) (Administration)'. The toolbar contains icons for 'New', 'Reports', 'Web Site', 'About', 'Address', 'Resources Organization', 'Bookings', and 'Charges'. A 'Quick Find ...' search box is located on the right side of the toolbar.

The main content area is divided into three panes:

- Left Pane (Favorites):** A tree view showing a hierarchy of folders and items under 'Programming Techniques', including 'Advanced Instance Lists', 'SubTypes', 'Using LANSAs Weblets', 'Switching', 'Debug using Trace', 'End User Help (F1)', 'Event Handling and Prom', 'Using Code Tables', 'View iSeries Spool Files', 'Mini Filter (Simple)', 'AJAX', 'Mini Filter (Multiple)', 'Mini Filter (Multiple 2)', 'Modify VLF at Runtime', and 'Text editor examples'.
- Middle Pane (About this example):** A text area containing the following text:

About this example

Instance lists may contain different relationships. This example uses the CUSTOMER, SUPPLIER and SHIP hierarchical way. They are loosely related.

To use this example click the Create button and search for the name "ACME".

The resulting instance list will find objects that are loosely related only by the fact that they contain the name "ACME".

As you click on objects in the instance list, information that is appropriate for the current object will be displayed.

Instance lists containing mixed peer relationships can be accessed in a "Google" style to information need to create.

At the bottom of this pane is a button labeled 'Create Example Instance List'.
- Right Pane (Instance List):** A table with columns 'Id', 'T...', and 'Name'. The table is highlighted with a red border. Below the table are 'Accounts', 'Bookings', 'Claims', and 'Contacts' icons, a scroll bar, and a 'Clear List' button.

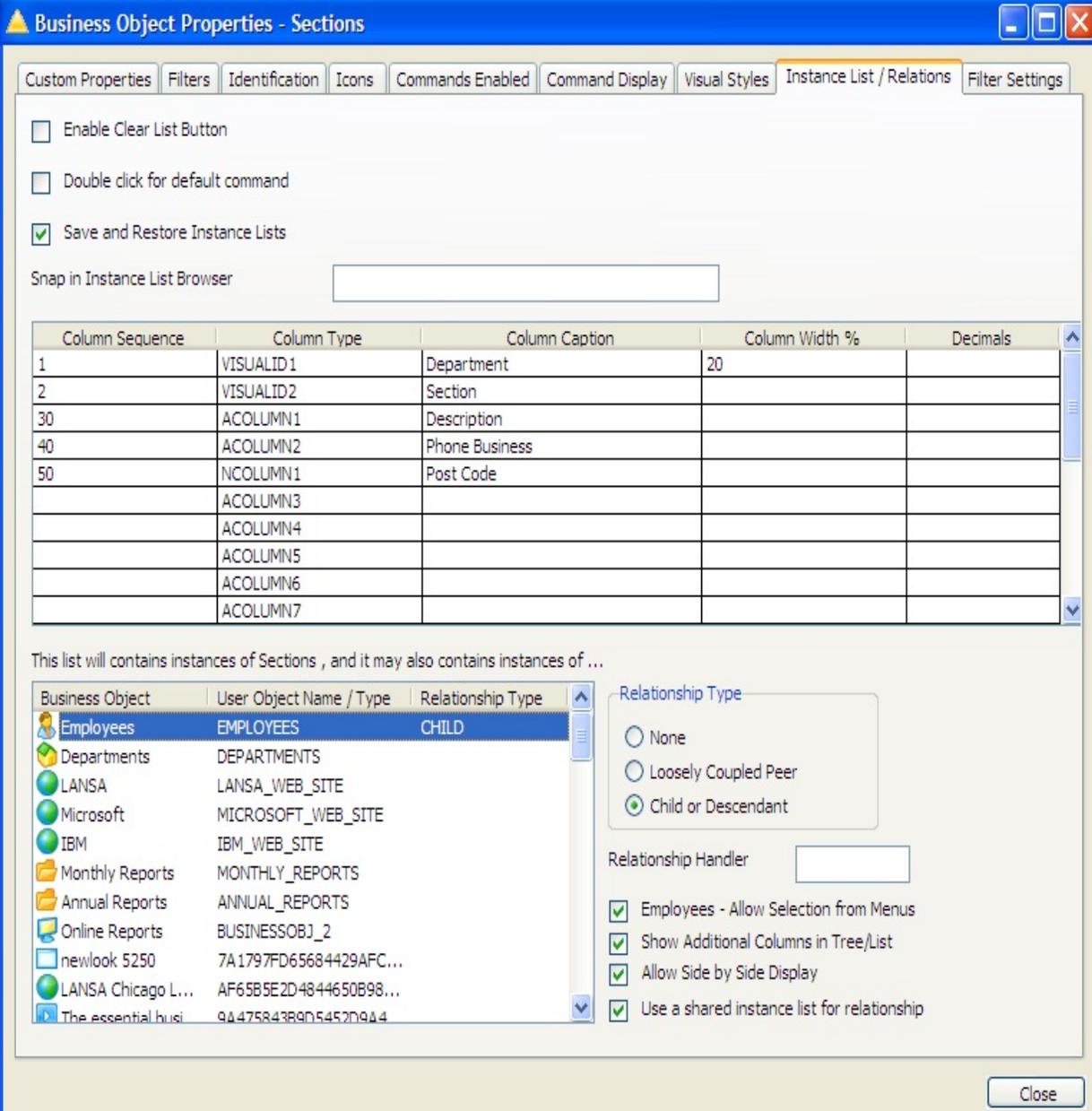
The status bar at the bottom of the window shows 'Messages', 'Ready', 'Local', 'ENG', 'VLFPGMLIB', '21/08/11', and '17:07'.

Id	T...	Name
A26273	S..	Acme Supermarkets
A34567	S..	ACME Eng.&Mfg.Corp
A63738	C..	Acme Engineering
DHG-46476-2	S..	ACME Laboratories
DHX-45455-7	S..	Electric Corporation - ...
Q26262	C..	ACME Nursery and Fl...
R83838	S..	Micro ACME Corpora...
Z17273	C..	Acme Plant and Equi...

Defining the Object Relationships

Display the Instance List/Relations tab in the properties of a business object to see the relationship. To see the relationship details, select an object in the list of available business objects.

In this example you can see that an EMPLOYEE is defined as a child of a SECTION:



The screenshot shows the 'Business Object Properties - Sections' dialog box, specifically the 'Instance List / Relations' tab. The dialog has several tabs: Custom Properties, Filters, Identification, Icons, Commands Enabled, Command Display, Visual Styles, Instance List / Relations, and Filter Settings. The 'Instance List / Relations' tab is active, showing a list of columns and a list of business objects.

The columns table is as follows:

Column Sequence	Column Type	Column Caption	Column Width %	Decimals
1	VISUALID1	Department	20	
2	VISUALID2	Section		
30	ACOLUMN1	Description		
40	ACOLUMN2	Phone Business		
50	NCOLUMN1	Post Code		
	ACOLUMN3			
	ACOLUMN4			
	ACOLUMN5			
	ACOLUMN6			
	ACOLUMN7			

Below the columns table, there is a text box for 'Snap in Instance List Browser' and a description: 'This list will contains instances of Sections , and it may also contains instances of ...'.

The business objects list is as follows:

Business Object	User Object Name / Type	Relationship Type
Employees	EMPLOYEES	CHILD
Departments	DEPARTMENTS	
LANSAs	LANSAs_WEB_SITE	
Microsoft	MICROSOFT_WEB_SITE	
IBM	IBM_WEB_SITE	
Monthly Reports	MONTHLY_REPORTS	
Annual Reports	ANNUAL_REPORTS	
Online Reports	BUSINESSOBJ_2	
newlook 5250	7A1797FD65684429AFC...	
LANSAs Chicago L...	AF65B5E2D4844650B98...	
The.essential.busi	9447584389D5452D944	

The 'Relationship Type' section shows three radio buttons: 'None', 'Loosely Coupled Peer', and 'Child or Descendant'. The 'Child or Descendant' option is selected.

The 'Relationship Handler' section has a text box and four checked checkboxes: 'Employees - Allow Selection from Menus', 'Show Additional Columns in Tree/List', 'Allow Side by Side Display', and 'Use a shared instance list for relationship'.

A 'Close' button is located at the bottom right of the dialog.

If you are using parent-child instance lists, you must use VISUALID1.

Do not set VISUALID1's Column Sequence to zero.

The relationship options define settings that impact the relationship and how the associated instance list is to be processed and displayed:

[Allow Selection from Navigation Pane](#)

[Show Additional Columns](#)

[Allow Side by Side Display](#)

[Use a Shared Instance List for Relationships](#)

Also see:

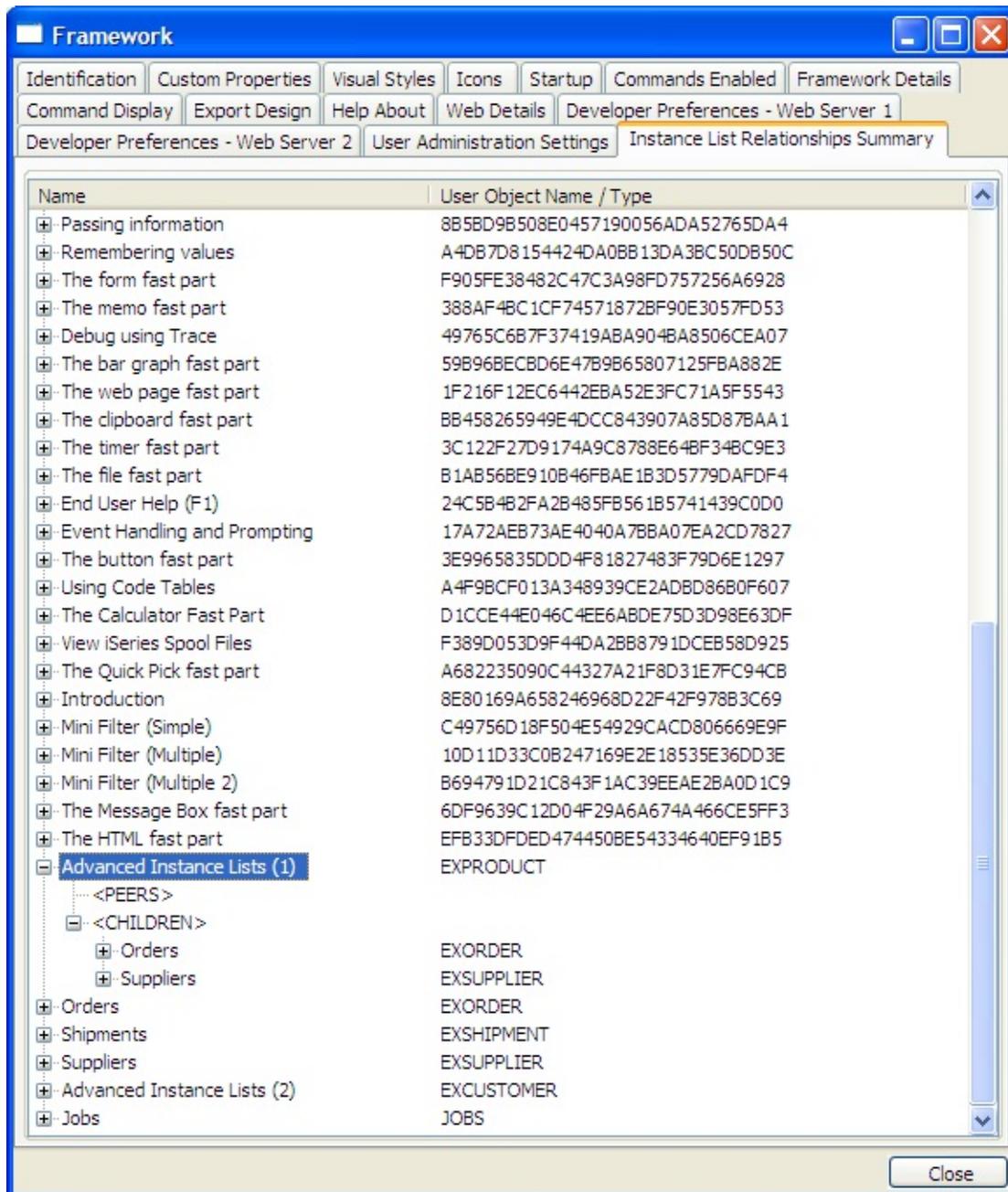
[Relationship Type](#)

[Relationship Handler](#)

Reviewing the Instance List Relationships Holistically

The (Framework) -> (Properties) -> Relationships tab shows the relationships between business objects across the entire Framework.

By expanding the Advanced Instance List business object (EXPRODUCT) we see that it has no 'Peer' objects and two 'Child' objects Orders and Suppliers.

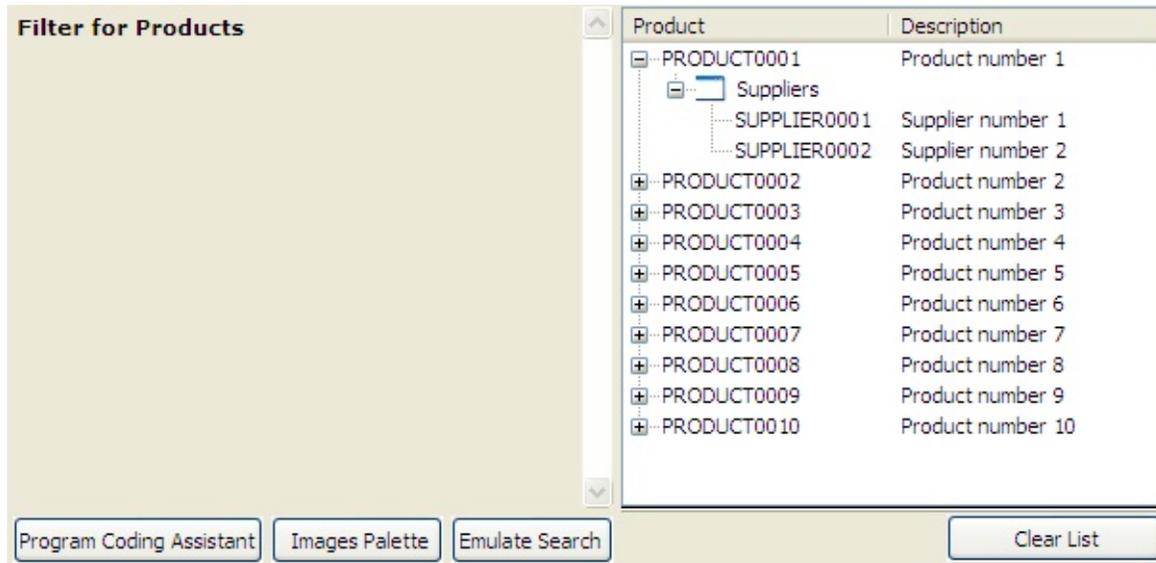


The screenshot shows a window titled "Framework" with a "Relationships Summary" tab selected. The window displays a list of business objects and their relationships. The "Advanced Instance Lists (1)" section is expanded, showing the following data:

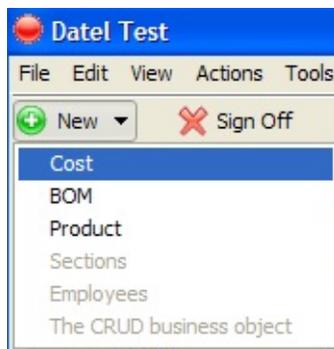
Name	User Object Name / Type
<PEERS>	
<CHILDREN>	
Orders	EXORDER
Suppliers	EXSUPPLIER
Orders	EXORDER
Shipments	EXSHIPMENT
Suppliers	EXSUPPLIER
Advanced Instance Lists (2)	EXCUSTOMER
Jobs	JOBS

Prototyping and Displaying Instance Lists

The standard shipped instance list browser understands that when child objects are involved, it needs to produce a tree structure rather than a flat list:



As you move up and down the instance list the various command handlers for the for the different types of objects appear according to the current selection. Equally, depending on the type of the object, commands and pop-up menus are enabled and disabled. In some contexts a command is qualified. For example, if you click on a COST and then use the "New" button on the menu bar, it will ask you to qualify what type of object:



Equally, if you display the pop-up menu over the instance list it will ask you the same thing:

Product	Description
PRODUCT0001	Mock Product number 1
BOMs	
BOM0001	Mock BOM number 1
BOM0002	Mock BOM number 2
Costs	
COST0003	Mock Cost number 3
COST0004	Mock Cost number 4
PRODUCT0002	Mock Product number 2
PRODUCT0003	Mock Product number 3
PRODUCT0004	Mock Product number 4
PRODUCT0005	Mock Product number 5
PRODUCT0006	Mock Product number 6

A context menu is open over the 'COST0003' entry, showing the following options:

- New
 - Product
 - BOM
 - Cost
- Notes
 - BOM
 - Cost
- Details
 - Cost
- Position
 - number 5

Adding Entries to the Instance List

The different types of objects may be added into the instance list in two different ways:

By the filter. The AddToList method has a new optional BusinessObjectType() parameter that allows filters to add different business object types to the same instance list. If passed as blanks/nulls or not specified this parameter defaults to the business object that owns the filter.

By a relationship handler. In a CHILD relationship only you can optionally specify a relationship handler. This is either an RDML function, or a reusable part, that is called to dynamically "expand" the relationship between a parent and child object. By doing this you can improve filter performance by only adding root or parent objects to the instance list initially.

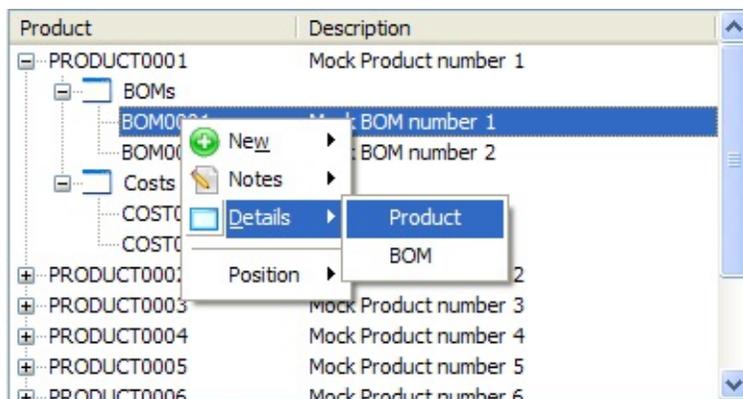
For a sample of such a function see:

- [Sample Relationship Handler Function](#)
- [Sample Relationship Reusable Part](#)

When objects exist in parent/child relationships there is assumed to be an inherent hierarchical relationship in the way that their programmatic identifiers are used in the instance list.

For example, if Product used AKey1(#Product_Number) to uniquely identify itself, the children BOMs and Costs are also expected to include the same (and correct) product number as their AKey1 values. They would both also use additional AKeyN and NKeyN values as well to uniquely identify themselves.

Since the instance list in this example now contains mixed objects (ie: Products, BOMs and Costs) there are some new considerations. For example, if the user right clicks on a BOM but elects to display the Details of the Product:



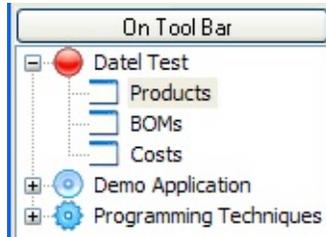
The Product Details command handler is invoked and displayed.

When it gets the current instance list entry it will actually get the currently selected BOM instance list entry. The program can tell this because the `GetCurrentInstance` method now optionally returns the `BusinessObjectType` value of the instance.

Most likely the Product Details command handler does not need to know the specific `BusinessObjectType` because the inherent hierarchical relationship in the programmatic identifiers used for Products, BOMs and Costs all put the product number in `AKey1 ...` which is all that the Product Details command handler really needs to know.

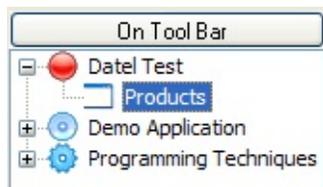
Hiding Objects in Navigation Pane

By default objects specified as child objects are still allowed to be selected from the navigation pane in their own right:



Depending up on the end-user's perception of what the child objects are you may or may not choose to make the disappear from the navigation pane.

To stop them appearing in the navigation panes (so that they are only visible as children of the parent object) go back to the Instance List/Relationship tab for the parent business object and uncheck the "Allow Selection from Navigation Pane" check boxes for both child objects. The child objects are hidden:

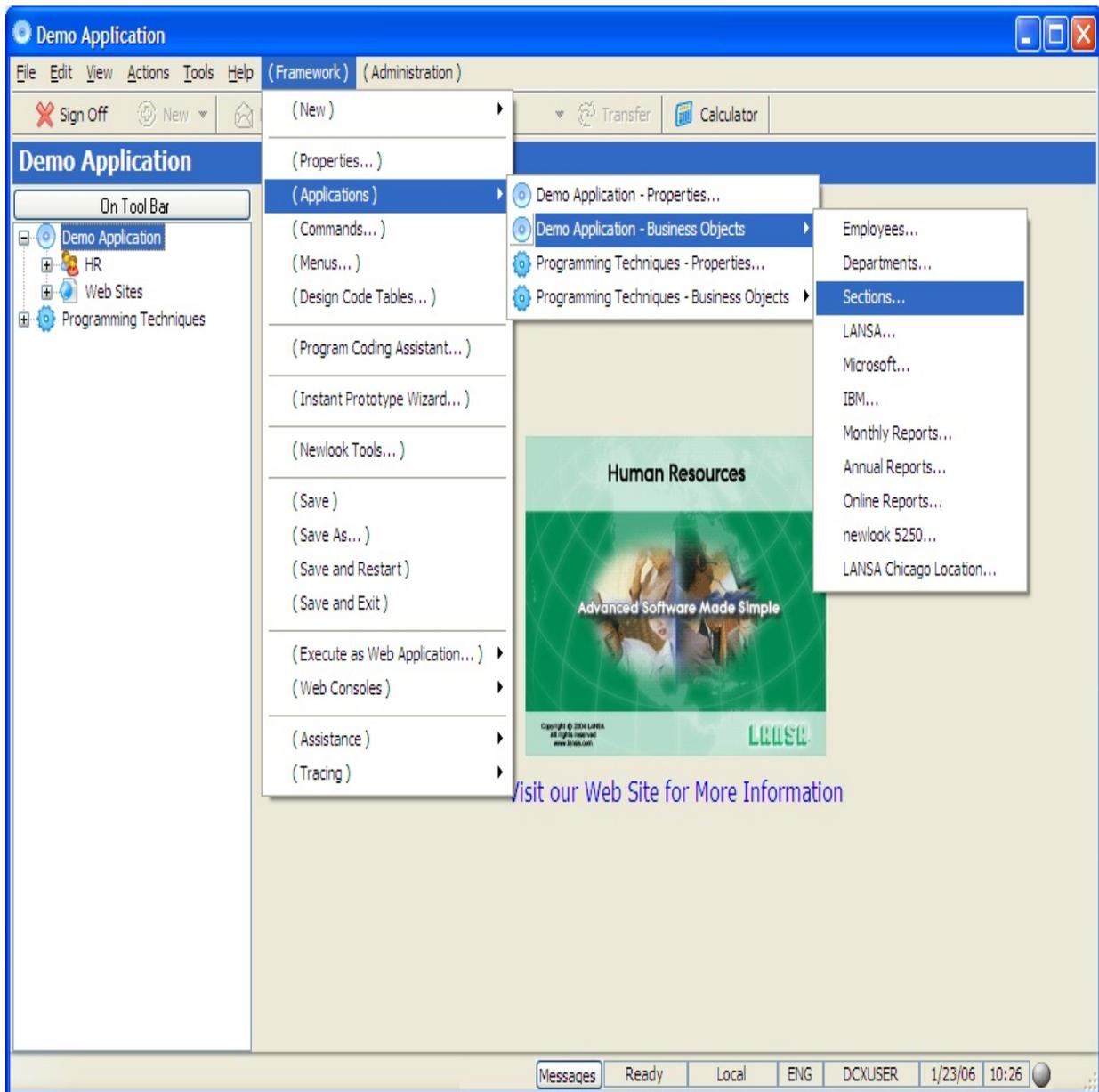


If you do leave them on the navigation pane, they continue to function as discrete business objects in their own right and have their own individual instance lists.

Work with Hidden Child Objects

To work with the properties of a child object which is not visible in the navigation pane you can:

- Double-click on any instance list entry related to the business object in question, or
- Select the (Applications...) option of the (Framework) menu, then select the application to which the object belongs to and then select it from the list of business objects:



Changing the order of child business objects in the instance list tree

When business object relationships are defined they appear in the instance list tree in the order they were defined. Currently there is no automatic facility to change the display order. If you need to change the order, do this:

Open your main XML (eg: VF_SY001_System.XML) with Notepad, maybe saving a copy first. Look for the relationship definition blocks. They look like this:

```
<OBJECT TYPE="RELATION" CLASS="VF_FP024" ID="whatever" >  
<.....>  
<.....>  
</OBJECT>
```

They will be nested inside a VF_FP003 block like this:

```
<OBJECT TYPE="BUSINESS_OBJECT" CLASS="VF_FP003"  
ID="whatever" >  
<.....>  
<.....>  
<OBJECT TYPE="RELATION" CLASS="VF_FP024" ID="whatever" >  
<.....>  
<.....>  
</OBJECT>  
  
<OBJECT TYPE="RELATION" CLASS="VF_FP024" ID="whatever" >  
<.....>  
<.....>  
</OBJECT>  
<.....>  
<.....>  
</OBJECT>
```

Rearrange the complete VF_FP024 blocks inside the VF_FP003 blocks into the desired order then save changes and restart VLF.

Things to Watch Out For

Relationships between business objects are purely visual and may be only vaguely conceptual. It is important to understand that the object relationship feature is not designed to be a data or database modeling facility.

Creating relationships based on the database design can lead to common errors you should try to avoid:

Starting Too High

Sometimes designers structure the trees like database table structures, but start too high in the structure forcing the end-user to drill-down too many levels. For example, they might structure a tree like this to reflect DBMS table relationships:



If the end-users are not really concerned with the Company and Division level then their presence in the tree makes Order navigation needlessly complex.

In a situation like this the Company and Division selection should be done by the filter and no tree structure. A simple list of orders would be much more appropriate and quicker for end-users to understand and use:

- Order
- Order
- Order
- Order
- Order
- Order

Going Down Too Low

Sometimes designers again use a database structure and go too low in the structure and break up the information into needlessly small fragments. For example, they might structure a tree like this:

```
Order
  Line Item 1
  Line Item 2
  Line Item 3
  Line Item 4
Order
  Line Item 1
  Line Item 2
```

This structure forces users to manipulate line items one by one, when working with all the line items in an order would be more efficient. So a simple list of orders would be much more effective for end-users to understand and use:

```
Order
Order
Order
Order
```

With a command tab for items that allow all the line items associated with the order to be viewed and manipulated in, for example, a single grid would probably be more appropriate.

Application Tracing for Relationship Handlers

If you are using relationship handlers 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.

Sample Relationship Handler Function

```
*
=====
* Component   : DF_PROC/DFREL01
* Type       : Function
* Disclaimer  : The following material is supplied as
* sample material only. No warranty concerning this
* material or its use in any way whatsoever is
* expressed or implied
*
=====
FUNCTION OPTIONS(*DIRECT *LIGHTUSAGE *MLOPTIMIZE)
RCV_LIST(#VIS_LIST #PID_LIST #COL1_LIST #COL2_LIST
#COL3_LIST #COL4_LIST #COL5_LIST #COL6_LIST #COL7_LIST
#COL8_LIST #COL9_LIST #COLA_LIST)
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL1)

* The Employee group

GROUP_BY NAME(#EMPLOYEE) FIELDS(#EMPNO #GIVENAME
#SURNAME #PHONEBUS #ADDRESS1 #ADDRESS2 #ADDRESS3
#POSTCODE #PHONEHME #DEPARTMENT #SECTION)

* Clear all keys and additional columns

EXECUTE SUBROUTINE(CLEARKEYS)
EXECUTE SUBROUTINE(CLEARCOLS)

* Field #SRC_TYPE (Source Business Object Type) is
* the source object of the relationship so switch
* on that initially .....

CASE OF_FIELD(#SRC_TYPE)

* Expand Sections in a Department/Organization

WHEN VALUE_IS(= DEM_ORG')
EXECUTE SUBROUTINE(ORGSEC)
```

* Expand Employees in a Department/Section

```
WHEN VALUE_IS(= DEM_ORG_SEC')  
EXECUTE SUBROUTINE(SECEMP)
```

OTHERWISE

```
ABORT MSGTXT('Unknown source business object type encountered by  
function DFREL01')
```

ENDCASE

* Finished ... always use a return command

RETURN

* Return the sections in a department/organization

```
SUBROUTINE NAME(ORGSEC)
```

```
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)
```

* We can now read all the EMPLOYEES in the specified

* department/section and add them to the instance list

```
SELECT FIELDS(*ALL) FROM_FILE(SECTAB)  
WITH_KEY(#DEPARTMENT)
```

```
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMs(1 #DEPARTMENT)
```

```
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMs(2 #SECTION)
```

```
EXECUTE SUBROUTINE(SETNCOL) WITH_PARMs(1 #SECPCODE)
```

```
EXECUTE SUBROUTINE(SETACOL) WITH_PARMs(1 #SECADDR1)
```

```
EXECUTE SUBROUTINE(SETACOL) WITH_PARMs(2 #SECADDR2)
```

```
EXECUTE SUBROUTINE(SETACOL) WITH_PARMs(3 #SECADDR3)
```

```
EXECUTE SUBROUTINE(SETACOL) WITH_PARMs(4 #SECPHBUS)
```

```
EXECUTE SUBROUTINE(ADDTOLIST)
WITH_PARMS('DEM_ORG_SEC' #SECDESC #SECTION)
ENDSELECT
```

* Finished

```
ENDROUTINE
```

```
SUBROUTINE NAME(SECEMP)
```

```
CHANGE FIELD(#DEPARTMENT) TO(#SRC_AK1)
CHANGE FIELD(#SECTION) TO(#SRC_AK2)
```

```
SELECT FIELDS(#EMPLOYEE) FROM_FILE(PSLMST1)
WITH_KEY(#DEPARTMENT #SECTION)
```

```
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMS(1 #DEPARTMENT)
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMS(2 #SECTION)
EXECUTE SUBROUTINE(SETAKEY) WITH_PARMS(3 #EMPNO)
```

```
EXECUTE SUBROUTINE(SETNCOL) WITH_PARMS(1 #POSTCODE)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(1 #ADDRESS1)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(2 #ADDRESS2)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(3 #ADDRESS3)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(4 #PHONEBUS)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(5 #PHONEHME)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(6 #DEPARTMENT)
EXECUTE SUBROUTINE(SETACOL) WITH_PARMS(7 #SECTION)
```

```
USE BUILTIN(TCONCAT) WITH_ARGS(#SURNAME ',' #GIVENAME)
TO_GET(#FULLNAME)
```

```
EXECUTE SUBROUTINE(ADDTOLIST)
WITH_PARMS('DEM_ORG_SEC_EMP' #FULLNAME #EMPNO)
```

```
ENDSELECT
```

* Finished

```
ENDROUTINE
```

```
*
```

```
=====
```

```
INCLUDE PROCESS(*DIRECT) FUNCTION(VFREL2)
```

Note: To add values to Date/Time additional columns use the SETDCOL subroutine in the same way that the SETACOL subroutine is used to add values to alphanumeric additional columns.

To convert the values in date/Time fields to ISO format (required for Date/Time additional columns) see the [Date/Time Additional Column Programming Example](#).

Sample Relationship Reusable Part

```
*
=====
*
* Component   : DF_T3507
* Type       : Reusable Component
* Ancestor   : VF_AC023
* May be used with both VLF-WEB and VLF-WIN versions
*
=====

Function Options(*DIRECT)
begin_com role(*EXTENDS #VF_AC023)

mthroutine name(uAddEntriesFor) options(*REDEFINE)

CASE OF_FIELD(#SourceType)

WHEN VALUE_IS('= DEM_ORG')
* Expand Sections in a Department/Organization

#DEPARTMENT := #AKEY1
SELECT FIELDS(*ALL) FROM_FILE(SECTAB)
WITH_KEY(#DEPARTMENT)

Signal uAddListItem AKey1(#DEPARTMENT) AKey2(#SECTION)
VisualId1(#SECDISC) VisualId2(#SECTION) AColumn1( #SECADDR1)
AColumn2(#SECADDR2) AColumn3(#SECADDR3)
AColumn4(#SECPHBUS) NColumn1(#SECPCODE)
BusinessObjectType(#TargetType)

ENDSELECT

WHEN VALUE_IS('= DEM_ORG_SEC')
* Expand Employees in a Department/Section
```

```
CHANGE FIELD(#DEPARTMENT) TO(#AKey1)
CHANGE FIELD(#SECTION) TO(#AKey2)
```

```
SELECT FIELDS(*ALL) FROM_FILE(PSLMST1)
WITH_KEY(#DEPARTMENT #SECTION)
```

```
Signal uAddListItem AKey1(#DEPARTMENT) AKey2(#SECTION)
AKey3(#EMPNO) VisualId1((#Givenname + ' ' + #Surname))
VisualId2(#EMPNO) AColumn1(#ADDRESS1) AColumn2(#ADDRESS2)
AColumn3(#ADDRESS3) AColumn4(#PHONEBUS)
AColumn5(#PHONEHME) AColumn6(#DEPARTMENT)
AColumn7(#SECTION) NColumn1(#POSTCODE)
BusinessObjectType(#TargetType)
```

```
endselect
```

```
OTHERWISE
```

```
ABORT MSGTXT('Unknown source business object type encountered by
component DF_T3507')
```

```
ENDCASE
```

```
endroutine
```

```
End_Com
```

Physical Instance Lists

As a developer it may be useful to understand how the actual instance list data content is physically stored and processed.

Some basics of the physical side of instance lists (also see [Programmatic Identifiers](#) earlier in this section):

- An instance list entry is defined and accessed by a programmatic identifier made of up to 5 alphanumeric and 5 numeric values. These are usually referred to as AKey1 -> AKey5 and NKey1 -> NKey5.
- The composite key that uniquely defines an instance entry is composed of AKey1-NKey1-AKey2-NKey2-AKey3-NKey3-AKey4-NKey5-AKey5-NKey5.
- Every instance list entry has this 10-part key, even when you do not specify all the keys. For example, when adding SECTIONS to the example instance list, only AKey1(#DEPARTMENT) and AKey2(#SECTION) are normally specified. When this is done, AKey3(' ') -> AKey5(' ') and NKey1(0) -> NKey5(0) are used as parameter default values to form the full 10-part key.
- The preceding point means that if you really want to use a blank AKeyn() values or zero NKeyn() values as part of a real key then it may be problematic. In such cases you should consider using something like AKey4('<BLANK>') and/or NKey2(-9999) to logically represent blank or zero as real non-blank and non-zero key values.
- The AKeyn() and NKeyn() values are purely conceptual. You can compose them in any way you feel is appropriate. For example, if you have a need for more than 5 AKeyn() keys, concatenate 2 or more of them together into a single AKeyn() value. For example the SECTIONS-EMPLOYEES instance list could have been structured with this key usage protocol - AKey1 is (#DEPARTMENT + "." + #SECTION) and AKey2 is (#EMPNO).

In the shipped SECTIONS business object the key usage protocol is defined as AKey1 = Department Code and AKey2 = Section Code.

In the shipped EMPLOYEES business object the key usage protocol as AKey1 = Department Code, AKey2 = Section Code and AKey3 = Employee Number.

This means there is a structured key relationship between a parent SECTION and child EMPLOYEE.

If you are trying to visualize the physically mixed SECTIONS/EMPLOYEES instance list in your mind, imagine the entries looking something like this:

Business Object Type	AKey1	AKey2	AKey3	Visual ID1	Visual ID2
SECTIONS	ADM	01		ADM	01
EMPLOYEES	ADM	01	A1001	A1001	BEN JONES
EMPLOYEES	ADM	01	A1012	A1012	PATRICK PAUL
SECTIONS	ADM	02		ADM	02
EMPLOYEES	ADM	02	A0090	A0090	FRED BLOOGS
EMPLOYEES	ADM	02	A1014	A1014	JOHN MOORE
SECTIONS	LEG	01		LEG	01
EMPLOYEES	LEG	01	A1019	A1019	CHARLES DICKENS
etc					

This key structure relationship between parent SECTIONS and child EMPLOYEES is absolutely vital to being able to process instance lists sensibly, and to displaying them in Visual LANSA tree controls. Also see [Planning parent-child relationships](#).

There are some things about this physical instance list that are worth noting:

- [You can add the entries to the instance list in any order](#)
- [You can dynamically add children to an instance list "on demand"](#)
- [When a child is in the instance list it must have a parent](#)
- [You can dynamically update individual entries in a parent-child instance list without collapsing the visual tree](#)
- [You can dynamically delete individual entries in a parent-child instance list without collapsing the visual tree](#)
- [You can dynamically add individual entries in a parent-child instance list without collapsing the visual tree](#)

Planning parent-child relationships

When setting the key structures for parent-child relationships - it can be helpful to fill in a table like the following example:

Here a SECTION may contain EMPLOYEES.

An EMPLOYEE may have associated SKILLS.

An EMPLOYEE may have associated DOCUMENTS.

BO	ROLE	AKey1	NKey1	AKey2
SECTION	PARENT	#DEPARTMENT	Implicitly used by parent	#SECTION
EMPLOYEE	CHILD (of SECTION)	Must be same as parent's #DEPARTMENT value	Implicitly used by parent - so must have same default value.	Must be same as parent's #SECTION value
SKILL	CHILD (of EMPLOYEE) GRANDCHILD (of SECTION)	Must be same as parent and grandparent's #DEPARTMENT value	Implicitly used by parent and grandparent - so must have same default value.	Must be same as parent and grandparent's #SECTION value
DOCUMENT	CHILD (of EMPLOYEE) GRANDCHILD (of SECTION)	Must be same as parent and grandparent's #DEPARTMENT value	Implicitly used by parent and grandparent - so must have same default value.	Must be same as parent and grandparent's #SECTION value

The parent-child association rules that you must follow are:

- Any child business object must have exactly the same keys as its parent plus one or more keys to uniquely identify itself.
- A child business objects can use the free keys not used by its parent. They can only start immediately after the last parent key used (real or implicit) working from left to right in the key structure.

You might also consider condensing key structures when your theoretical planning is finished – the keys are programmatic and not visible to end-users.

For example: by choosing #DEPARTMENT + “-“ + #SECTION as the AKey1 value in the table above you could condense it towards the left by two columns - freeing up a considerable amount of key space.

You can add the entries to the instance list in any order

In the example above you could add all the EMPLOYEES first in any order and then add the SECTIONS at the end in any order. The order you add entries does not matter.

You can dynamically add children to an instance list "on demand"

In this example you could just add the three SECTIONS like this:

Business Object Type	AKey1	AKey2	AKey3	Visual ID1	Visual ID2
SECTIONS	ADM	01		ADM	01
SECTIONS	ADM	02		ADM	02
SECTIONS	LEG	01		LEG	01

When the user goes to expand the section ADM-01 (say) in the visual tree a check is made to see if you have specified a Relationship Handler for the relationship between SECTIONS and EMPLOYEES.

If you have, then it is called to "expand" the relationship between SECTION ADM-01 and the EMPLOYEES that work in the section. After it has been called, the physical instance list would look like this:

Business Object Type	AKey1	AKey2	AKey3	Visual ID1	Visual ID2
SECTIONS	ADM	01		ADM	01
EMPLOYEES	ADM	01	A1001	A1001	BEN JONES
EMPLOYEES	ADM	01	A1012	A1012	PATRICK PAUL
SECTIONS	ADM	02		ADM	02
SECTIONS	LEG	01		LEG	01

At the end of this section is an example of a RDML function used to handle the expansion if the relationship between SECTIONS and an EMPLOYEES.

If you create a relationship handler to dynamically expand entries in an instance list then you need to identify it to the Framework on the Instance List /

Relations tab as the Relationship Handler here:

Relationship Type

None

Loosely Coupled Peer

Child or Descendant

Relationship Handler

Employees - Allow Selection from Menus

Show Additional Columns in Tree/List

Allow Side by Side Display

Use a shared instance list for relationship

When a child is in the instance list it must have a parent

If a child object is in the instance it must have a parent in the instance list that can be associated with that has the same structural key.

If one does not exist an error message will be displayed at some stage, probably when the instance list needs to be visualized as tree control.

For example, if an EMPLOYEE, identified by the fully key ADM-0-01-0-A1001, needs to be visualized in a Visual LANSA tree then a search is made for its parent SECTION.

This is done by looking, using the AKey1-NKey1-AKey2-NKey2-AKey3-NKey3-AKey4-NKey5-AKey5-NKey5 structured key, first for SECTIONS-ADM (which would not found), then SECTIONS-ADM-0 (which would also not be found) then SECTIONS-ADM-0-01, which would be found, thus identifying the parent correctly.

You can dynamically update individual entries in a parent-child instance list without collapsing the visual tree

To do this use `#AvListManager.BeginListUpdate Mode(DYNAMIC)`, as in these examples:

To change SECTION ADM-02 to have a VisualID2 of hello:

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)
```

```
Invoke Method(#avListManager.UpdateListEntryData) AKey1('ADM')  
AKey2('02') VisualID2('hello') BusinessObjectType(SECTIONS)
```

```
Invoke Method(#avListManager.EndListUpdate)
```

To change employee number A1012's VisualID2 to FREDDO FROG:

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)
```

```
Invoke Method(#avListManager.UpdateListEntryData) Visualid2('FREDDO  
FROG') Akey1('ADM') Akey2('01') Akey3('A1012')  
BusinessObjectType(EMPLOYEES)
```

```
Invoke Method(#avListManager.EndListUpdate)
```

Also see [Updating and Deleting Instance List Entries](#)

You can dynamically delete individual entries in a parent-child instance list without collapsing the visual tree

To do this use `#AvListManager.BeginListUpdate Mode(DYNAMIC)`, as in these examples:

To delete employee number A1012 from the instance list:

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)
```

```
Invoke Method(#avListManager.RemoveFromList) Akey1('ADM')  
Akey2('01')  
    Akey3('A1012') BusinessObjectType(EMPLOYEES)
```

```
Invoke Method(#avListManager.EndListUpdate)
```

To delete SECTION ADM-02 and its child EMPLOYEES from the instance list

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)
```

```
Invoke Method(#avListManager.RemoveFromList) AKey1('ADM')  
AKey2('02')  
    BusinessObjectType(SECTIONS)
```

```
Invoke Method(#avListManager.EndListUpdate)
```

Also see [Updating and Deleting Instance List Entries](#)

You can dynamically add individual entries in a parent-child instance list without collapsing the visual tree

To do this use `#AvListManager.BeginListUpdate Mode(DYNAMIC)`, as in these examples:

To add employee number A1012 to the instance list:

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)

Invoke Method(#avListManager.AddtoList) Visualid1('A1012')
Visualid2('PATRICK PAUL') Akey1('ADM') Akey2('01') Akey3('A1012')
AColumn1('8217-436474') AColumn2('121 MAIN STREET')
nColumn1(4353) BusinessObjectType(EMPLOYEES)

Invoke Method(#avListManager.EndListUpdate)
```

To add SECTION ADM-99 to the instance list:

```
Invoke Method(#avListManager.BeginListUpdate) Mode(DYNAMIC)

Invoke Method(#avListManager.AddtoList) Akey1('ADM') Akey2('99')
VisualID1('ADM') VisualID2('99') AColumn1('DEMO SECTION')
AColumn2('1627-7484') NColumn1('3478') BusinessObjectType(SECTIONS)

Invoke Method(#avListManager.EndListUpdate)
```

Also see [Updating and Deleting Instance List Entries](#)

Instance List Tips and Techniques

Page-at-a-Time logic may indicate a conceptual problem with filters

Instance lists do not have to reflect the database that underpins them

Page-at-a-Time logic may indicate a conceptual problem with filters

You can use the classic "page at a time" logic with instance lists, but an over-reliance on it may indicate that your filters have not been designed properly.

The purpose of a filter is to give the end-user sufficient search power to rapidly produce short instance lists of exactly what they want to work with. If your end-users must always have to page to find the information they want to work with you should consider increasing the "zoom in" capability of your filters before attempting to implement complex paging techniques.

For example, if you are designing a Human Resources system, one of the essential pieces of analysis you need to do is to work out all the different ways that end-users would want to build list of employees. For example:

- By their names
- By their phone numbers
- By the office they work in
- By their current employment status
- By their sex
- By their birthdays
- By the date they started working for the company
- Those that have resigned but are still on file
- By their skills
- By their remuneration ranges
- By those without outstanding employment issues.
- Etc, etc

If you have a system with 5000 employees then simply blasting the personal file out onto the screen with some "page at a time" logic will not make the end-users life any easier.

Sometimes Framework designers take this one step further and let end-users define (and store) standard employee 'queries' that they can use over and over in filters to build instance lists.

For example, a HR person tasked to ensure employee satisfaction, may, on

every Monday morning, need to build a list of all the people who started work last week.

Equally another may need to build a list every Friday afternoon of people working at a certain remote location so as to phone them up to see if they have any special transportation requirements needed on the weekend.

Think about giving end-users the EXACT list of what they need rather than a large list with some accompanying "page at a time" scrolling logic.

Instance lists do not have to reflect the database that underpins them

Instance lists are conceptual. They often reflect the physical structure of the database table(s) that underpins them, but they don't have to.

In this simple SECTIONS-EMPLOYEES relationship used through these examples, imagine the visual impact of a section containing 2000 employees. In a case like this you might consider inventing a 4 different child employee business objects named EMPLOYEES_A_G, EMPLOYEES_H_M, EMPLOYEES_N_T, EMPLOYEES_U_END to split up the children alphabetically into 4 groups.

Here's another simple example. Imagine you had a single database file containing messages. Each message has a unique identifying 7 digit number. Each message also has a status, somewhat like an e-mail, of RECEIVED, READ, SENT and DELETED. Conceptually this might be arranged into an instance list like this:

Business Object	Type	AKey1	AKey2
RECEIVE		RECEIVE	
MESSAGE		RECEIVE	26272
MESSAGE		RECEIVE	63738
READ		READ	
MESSAGE		READ	73389
MESSAGE		READ	74584
MESAAGE		READ	73873
SENT		SENT	
MESSAGE		SENT	78383
MESSAGE		SENT	37383
DELETED		DELETE	

MESSAGE

DELETE 62727

Conceptually the instance looks something like an e-mail inbox, outbox and deleted items. It is not directly reflective of the underpinning database design.

Create Your Own Snap-in Instance List

If the shipped instance list browser will not do exactly what you need then you can always write your own. See [Optionally Create Your Own Instance List](#)

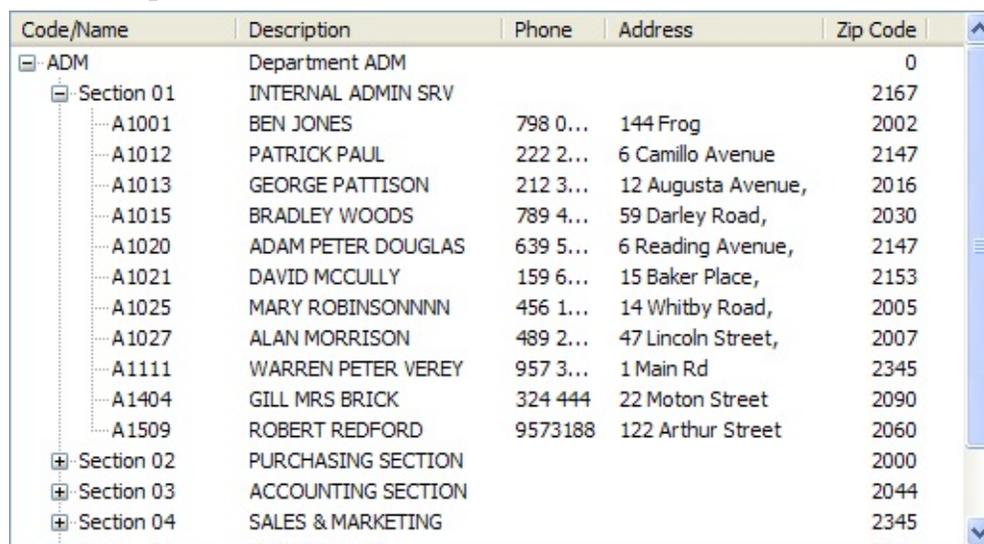
When using a snap-in instance list browser, the normal generic instance list layout details (columns, sequence, width, etc) on the "Instance List / Relations" tab have no meaning. You have full control of all these options from within your own snap in browser.

Your snap in browser can visualize the instance list in any way desired (list, grid, tree, etc).

Instance Lists – An Example Snap-In Instance List Browser

This example snap in instance list browser visualizes a SECTIONS-EMPLOYEES instance list as a 3 level tree.

It's method of displaying the instance list information has been significantly tailored to be different to the way that the standard shipped Instance List Browser presents the information:



Code/Name	Description	Phone	Address	Zip Code
ADM	Department ADM			0
Section 01	INTERNAL ADMIN SRV			2167
A1001	BEN JONES	798 0...	144 Frog	2002
A1012	PATRICK PAUL	222 2...	6 Camillo Avenue	2147
A1013	GEORGE PATTISON	212 3...	12 Augusta Avenue,	2016
A1015	BRADLEY WOODS	789 4...	59 Darley Road,	2030
A1020	ADAM PETER DOUGLAS	639 5...	6 Reading Avenue,	2147
A1021	DAVID MCCULLY	159 6...	15 Baker Place,	2153
A1025	MARY ROBINSONNNN	456 1...	14 Whitby Road,	2005
A1027	ALAN MORRISON	489 2...	47 Lincoln Street,	2007
A1111	WARREN PETER VEREY	957 3...	1 Main Rd	2345
A1404	GILL MRS BRICK	324 444	22 Moton Street	2090
A1509	ROBERT REDFORD	9573188	122 Arthur Street	2060
Section 02	PURCHASING SECTION			2000
Section 03	ACCOUNTING SECTION			2044
Section 04	SALES & MARKETING			2345

[Source for Snap-in Instance List Example](#)

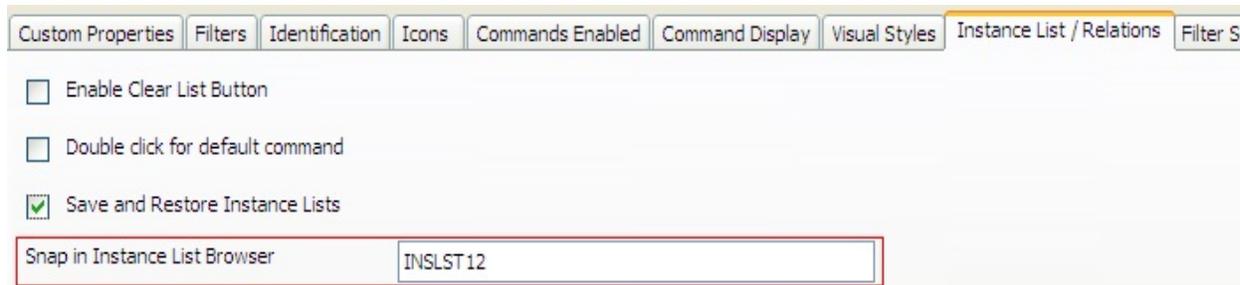
[Instance List with more than 10 alphanumeric and/or 10 numeric additional columns](#)

Instance List with more than 10 alphanumeric and/or 10 numeric additional columns

The Framework instance list can display up to 10 alphanumeric and/or 10 numeric additional columns in an instance list. If you want more columns you need to create your own snap-in instance list. There is no limit as to how many additional columns can be displayed by a snap-in instance list browser.

To create the instance list use the Code Assistant to create most of the required code for you. A working example is shipped in component DF_INST1.

Instance List Browsers are snapped in on the Business Object properties tab Instance List / Relations:



The simplest technique for putting an effectively infinite number of additional columns into the instance list is to use the **AColumn<symbolic name>(value)** and **NColumn<symbolic name>(value)** properties of the instance list manager.

(Note: This is actually an old technique that has disappeared over time because of the popularity of the newer AColumnN() and NColumnN() parameters on the AddtoList method. This used to be the only way that any additional columns could be used.)

In the following example filter following this is the salient RDMLX code:

```
* Make 16 alphanumeric columns with a variety of symbolic names such as  
BILL, MARY .. A16
```

```
set #avListManager AColumn<BILL>(#Char07 + BILL)  
set #avListManager AColumn<MARY>(#Char07 + MARY)  
set #avListManager AColumn<TOTAL>(#Char07 + TOTAL)  
set #avListManager AColumn<CUSTNO>(#Char07 + CUSTNO)  
set #avListManager AColumn<A5>(#Char07 + A5)  
set #avListManager AColumn<A6>(#Char07 + A6)  
set #avListManager AColumn<A7>(#Char07 + A7)  
set #avListManager AColumn<A8>(#Char07 + A8)
```

```
set #avListManager AColumn<A9>(#Char07 + A9)
set #avListManager AColumn<A10>(#Char07 + A10)
set #avListManager AColumn<A11>(#Char07 + A11)
set #avListManager AColumn<A12>(#Char07 + A12)
set #avListManager AColumn<A13>(#Char07 + A13)
set #avListManager AColumn<A14>(#Char07 + A14)
set #avListManager AColumn<A15>(#Char07 + A15)
set #avListManager AColumn<A16>(#Char07 + A16)
```

* Make 16 numeric columns with a variety of symbolic names such as BILL, MARY ... N16

```
set #avListManager nColumn<BILL>(#Zone07 + 1)
set #avListManager nColumn<MARY>(#Zone07 + 2)
set #avListManager nColumn<TOTAL>(#Zone07 + 3)
set #avListManager nColumn<CUSTNUM>(#Zone07 + 4)
set #avListManager nColumn<n5>(#Zone07 + 5)
set #avListManager nColumn<n6>(#Zone07 + 6)
set #avListManager nColumn<n7>(#Zone07 + 7)
set #avListManager nColumn<n8>(#Zone07 + 8)
set #avListManager nColumn<n9>(#Zone07 + 9)
set #avListManager nColumn<n10>(#Zone07 + 10)
set #avListManager nColumn<n11>(#Zone07 + 11)
set #avListManager nColumn<n12>(#Zone07 + 12)
set #avListManager nColumn<n13>(#Zone07 + 13)
set #avListManager nColumn<n14>(#Zone07 + 14)
set #avListManager nColumn<n15>(#Zone07 + 15)
set #avListManager nColumn<n16>(#Zone07 + 16)
```

* Add this and all 32 additional columns to the instance list

```
Invoke Method(#avListManager.AddtoList) Visualid1(#Char07)
Visualid2(#char07) Akey1(#char07)
```

As an example, the line `set #avListManager AColumn<CUSTNO>(#Char07 + CUSTNO)` creates an additional alphanumeric column symbolically named CUSTNO and sets it to contain the current value of field #CHAR07 concatenated with the string 'CUSTNO' .

In the associated snap in instant list manager this is the matching RDMLX code:

```
* Set the visual identifiers into the grid columns

#STD_TEXTS := #VisualID1
#STD_TEXT := #VisualID2

* Get the additional alphanumeric columns into the grid columns.
* Only 3 are visualized in this example, but all 16 could be display if required

#VF_ELXA01 := #AvListManager.Acolumn<BILL>
#VF_ELXA02 := #AvListManager.Acolumn<CUSTNO>
#VF_ELXA03 := #AvListManager.Acolumn<A16>

* Get the additional numeric columns into the grid columns.
* Only 3 are visualized in this example, but all 16 could be visualized if
required

#VF_ELXNK1 := #AvListManager.Ncolumn<BILL>
#VF_ELXNK2 := #AvListManager.Ncolumn<CUSTNUM>
#VF_ELXNK3 := #AvListManager.Ncolumn<N16>

* Add the information to the grid

Add_Entry #GRID
```

The line `#VF_ELXA02 := #AvListManager.Acolumn<CUSTNO>` shows the additional alphanumeric column named CUSTNO being copied in a grid column field #VF_ELXA02.

Command handlers can also access the additional columns using the same approach.

In high volume exchanges with a very significant number of additional columns there are possibly more "locked in" (ie: less generic and therefore faster) approaches that might be used to pass additional column information between a filters and an instance list browsers.

For example, just a single additional column may be used to pass a "row key" between the filter and the snap in instance list browser in the instance list. A unique scope(*Application) shared reusable VL component could be created to

manage the storage of and access to logical rows of information, possibly using a SPACE object for efficiency. Both the filter and instance list browser would communicate with it to exchange logical row details.

[Sample Filter using 34 columns \(16 alpha, 16 numeric + Visual IDs\)](#)

[Matching Snap in Instance List Browser](#)

Sample Filter using 34 columns (16 alpha, 16 numeric + Visual IDs)

```
BEGIN_COM ROLE(*EXTENDS #VF_AC007) HEIGHT(123)
WIDTH(216)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1) CAPTION('Add
to list ') DISPLAYPOSITION(1) LEFT(52) PARENT(#COM_OWNER)
TABPOSITION(1) TOP(28)
```

```
EVTROUTINE HANDLING(#PHBN_1.Click)
```

```
Define #Zone07 Reffld(#Date) Length(7) decimals(0) edit_code(4) Default(0)
Define #Char07 *char 7 To_Overlay(#Zone07)
```

```
Invoke Method(#avListManager.BeginListUpdate)
```

```
Invoke Method(#avListManager.ClearList)
```

```
Begin_Loop from(1) to(50) using(#Zone07)
```

* Make up 16 alphanumeric columns with a variety of symbolic names such as BILL, MARY ... A16

```
set #avListManager AColumn<BILL>(#Char07 + BILL)
set #avListManager AColumn<MARY>(#Char07 + MARY)
set #avListManager AColumn<TOTAL>(#Char07 + TOTAL)
set #avListManager AColumn<CUSTNO>(#Char07 + CUSTNO)
set #avListManager AColumn<A5>(#Char07 + A5)
set #avListManager AColumn<A6>(#Char07 + A6)
set #avListManager AColumn<A7>(#Char07 + A7)
set #avListManager AColumn<A8>(#Char07 + A8)
set #avListManager AColumn<A9>(#Char07 + A9)
set #avListManager AColumn<A10>(#Char07 + A10)
set #avListManager AColumn<A11>(#Char07 + A11)
set #avListManager AColumn<A12>(#Char07 + A12)
set #avListManager AColumn<A13>(#Char07 + A13)
```

```
set #avListManager AColumn<A14>(#Char07 + A14)
set #avListManager AColumn<A15>(#Char07 + A15)
set #avListManager AColumn<A16>(#Char07 + A16)
```

* Make up 16 numeric columns with a variety of symbolic names such as BILL, MARY ... N16

```
set #avListManager nColumn<BILL>(#Zone07 + 1)
set #avListManager nColumn<MARY>(#Zone07 + 2)
set #avListManager nColumn<TOTAL>(#Zone07 + 3)
set #avListManager nColumn<CUSTNUM>(#Zone07 + 4)
set #avListManager nColumn<n5>(#Zone07 + 5)
set #avListManager nColumn<n6>(#Zone07 + 6)
set #avListManager nColumn<n7>(#Zone07 + 7)
set #avListManager nColumn<n8>(#Zone07 + 8)
set #avListManager nColumn<n9>(#Zone07 + 9)
set #avListManager nColumn<n10>(#Zone07 + 10)
set #avListManager nColumn<n11>(#Zone07 + 11)
set #avListManager nColumn<n12>(#Zone07 + 12)
set #avListManager nColumn<n13>(#Zone07 + 13)
set #avListManager nColumn<n14>(#Zone07 + 14)
set #avListManager nColumn<n15>(#Zone07 + 15)
set #avListManager nColumn<n16>(#Zone07 + 16)
```

* Add this and all the additional columns to the instance list

```
Invoke Method(#avListManager.AddtoList) Visualid1(#Char07)
Visualid2(#char07) Akey1(#char07)
```

```
End_Loop
```

```
Invoke Method(#avListManager.EndListUpdate)
```

```
ENDROUTINE
```

```
End_Com
```

Matching Snap in Instance List Browser

```
BEGIN_COM ROLE(*EXTENDS #VF_AC012) HEIGHT(218)
HINT(*MTXTDF_INST1)
LAYOUTMANAGER(#ATTACHMENT_MANAGER) WIDTH(504)
```

* Basic attachment layout manager

```
DEFINE_COM CLASS(#PRIM_ATLM)
NAME(#ATTACHMENT_MANAGER)
```

* A grid display VID1,2 and 3 alpha columns and 3 numeric columns

```
DEFINE_COM CLASS(#PRIM_GRID) NAME(#grid)
COLUMNBUTTONHEIGHT(18) DISPLAYPOSITION(1) HEIGHT(218)
HINT(*MTXTDF_INST1) LEFT(0) PARENT(#COM_OWNER)
SELECTIONSTYLE(Multiple) SHOWLINES(False)
SHOWSELECTION(True) SHOWSORTARROW(True) TABPOSITION(1)
TABSTOP(False) TOP(0) WIDTH(504)
DEFINE_COM CLASS(#PRIM_ATLI)
NAME(#GRID_ATTACHMENT_ITEM) ATTACHMENT(Center)
MANAGE(#grid) PARENT(#ATTACHMENT_MANAGER)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_1)
CAPTION('VisualID1') CAPTIONTYPE(Caption) DISPLAYPOSITION(1)
PARENT(#grid) SORTONCLICK(True) SOURCE(#STD_TEXTS)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_2)
CAPTION('VisualID2') CAPTIONTYPE(Caption) DISPLAYPOSITION(2)
PARENT(#grid) SORTONCLICK(True) SOURCE(#STD_TEXT)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_3)
CAPTION('ACol-BILL') CAPTIONTYPE(Caption) DISPLAYPOSITION(3)
PARENT(#grid) SORTONCLICK(True) SOURCE(#VF_ELXA01)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_4)
CAPTION('ACol-CUSTNO') CAPTIONTYPE(Caption)
DISPLAYPOSITION(4) PARENT(#grid) SORTONCLICK(True)
SOURCE(#VF_ELXA02)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_5)
CAPTION('ACol-A16') CAPTIONTYPE(Caption) DISPLAYPOSITION(5)
PARENT(#grid) SORTONCLICK(True) SOURCE(#VF_ELXA03)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_6)
```

```
CAPTION('NCol-BILL') CAPTIONTYPE(Caption) DISPLAYPOSITION(6)
PARENT(#grid) SORTONCLICK(True) SOURCE(#VF_ELXNK1)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_7)
CAPTION('NCol-CUSTNUM') CAPTIONTYPE(Caption)
DISPLAYPOSITION(7) PARENT(#grid) SORTONCLICK(True)
SOURCE(#VF_ELXNK2)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_8)
CAPTION('NCol-N16') CAPTIONTYPE(Caption) DISPLAYPOSITION(8)
PARENT(#grid) SORTONCLICK(True) SOURCE(#VF_ELXNK3)
```

```
* -----
```

```
* Redefine the standard uClearInstanceList method
```

```
* -----
```

```
MthRoutine uClearInstanceList Options(*Redefine)
```

```
* Clear the visible sections grid of all entries
```

```
Clr_List #Grid
```

```
EndRoutine
```

```
* -----
```

```
* Redefine the standard uAddListEntry method
```

```
* -----
```

```
Mthroutine Name(uAddListEntry) Options(*Redefine)
```

```
* Set the visual identifiers
```

```
#STD_TEXTS := #VisualID1
```

```
#STD_TEXT := #VisualID2
```

```
* Get the additional alphanumeric columns.
```

```
* Only 3 are visualized, but all 16 could be shown if required
```

```
#VF_ELXA01 := #AvListManager.Acolumn<BILL>
```

```
#VF_ELXA02 := #AvListManager.Acolumn<CUSTNO>
```

```
#VF_ELXA03 := #AvListManager.Acolumn<A16>
```

- * Get the additional numeric columns.
- * Only 3 are visualized, but all 16 could be shown if required

```
#VF_ELXNK1 := #AvListManager.Ncolumn<BILL>  
#VF_ELXNK2 := #AvListManager.Ncolumn<CUSTNUM>  
#VF_ELXNK3 := #AvListManager.Ncolumn<N16>
```

- * Add the entry to the visible grid

```
Add_Entry #GRID
```

- * Finished

```
Endroutine
```

- * -----
* Handle selection of a section in the grid
* -----

```
EVTROUTINE HANDLING(#Grid.ItemGotSelection)  
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
```

- * Appropriate code needs to be added
- ```
ENDROUTINE
```

- \* -----  
\* Handle unselection of a section in the grid  
\* -----

```
EVTROUTINE HANDLING(#Grid.ItemLostSelection)
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
```

- \* Appropriate code needs to be added
- ```
ENDROUTINE
```

- * -----
* Handle focus of a section in the grid
* -----

```
EVTROUTINE HANDLING(#Grid.ItemGotFocus)
```

```
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
```

```
* Appropriate code needs to be added
```

```
ENDROUTINE
```

```
* -----
```

```
* Handle loss of focus of a section in the grid
```

```
* -----
```

```
EVTROUTINE HANDLING(#Grid.ItemLostFocus)
```

```
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
```

```
* Appropriate code needs to be added
```

```
ENDROUTINE
```

```
EVTROUTINE HANDLING(#Grid.ItemGotFocusAccept
```

```
#Grid.ItemGotSelectionAccept) Accept(#ACCEPT)
```

```
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
```

```
* Appropriate code needs to be added
```

```
Set Com(#ACCEPT) Value(TRUE)
```

```
ENDROUTINE
```

```
END_COM
```

Source for Snap-in Instance List Example

The example code is:

```
*
=====
* Component   : XXXXXXXXXXXX
* Type       : Reusable Part
* Description : Sample of a snap in instance list browser
* displaying an instance list as a tree control.
* Disclaimer  : The following material is supplied as
* sample material only. No warranty concerning this
* material or its use in any way whatsoever is
* expressed or implied
*
=====

BEGIN_COM ROLE(*EXTENDS #VF_AC012) HEIGHT(181)
LAYOUTMANAGER(#ATTACHMENT_MANAGER) WIDTH(482)

* Overall attachment layout manager

DEFINE_COM CLASS(#PRIM_ATLM)
NAME(#ATTACHMENT_MANAGER)

* The unlevelled tree

DEFINE_COM CLASS(#PRIM_TRVW) NAME(#VIS_Tree)
COLUMNBUTTONHEIGHT(18) DISPLAYPOSITION(1)
DRAGCOLUMNS(True) FULLROWSELECT(True) HEIGHT(181) LEFT(0)
MULTIPLESELECTSTYLE(SameLevel) PARENT(#COM_OWNER)
SELECTIONSTYLE(Multiple) TABPOSITION(1) TABSTOP(False) TOP(0)
VIEWSTYLE(UnLevelled) VISUALSTYLE(#VF_VS101) WIDTH(482)

* Attachment item for layout management

DEFINE_COM CLASS(#PRIM_ATLI)
NAME(#TREE_ATTACHMENT_ITEM) ATTACHMENT(Center)
```

MANAGE(#VIS_Tree) PARENT(#ATTACHMENT_MANAGER)

* Standard 2 fields for all levels in the tree

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#Tree_VID1)
CAPTION('Code/Name') CAPTIONTYPE(Caption) DISPLAYPOSITION(1)
LEVEL(1) PARENT(#VIS_Tree) SORTONCLICK(True)
SOURCE(#VF_ELXVI1) WIDTH(17)
```

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#Tree_VID2)
CAPTION('Description') CAPTIONTYPE(Caption) DISPLAYPOSITION(2)
LEVEL(1) PARENT(#VIS_Tree) SORTONCLICK(True)
SOURCE(#VF_ELXVI2) WIDTH(16)
```

* 2 sample additional alpha columns

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#Tree_acolumn1)
CAPTION('Phone') CAPTIONTYPE(Caption) DISPLAYPOSITION(3)
LEVEL(1) PARENT(#VIS_Tree) SORTONCLICK(True)
SOURCE(#VF_ELXCA1) WIDTH(13)
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#Tree_acolumn2)
CAPTION('Address') CAPTIONTYPE(Caption) DISPLAYPOSITION(4)
LEVEL(1) PARENT(#VIS_Tree) SORTONCLICK(True)
SOURCE(#VF_ELXCA2) WIDTH(14)
```

* 1 sample additional numeric column

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#Tree_Visncolumn1)
CAPTION('Zip Code') CAPTIONTYPE(Caption) COLUMNALIGN(Right)
DISPLAYPOSITION(5) LEVEL(1) PARENT(#VIS_Tree)
SORTONCLICK(True) SOURCE(#POSTCODE)
```

* Hidden columns to track AKey1() AKey2() AKey3() and
BusinessObjectType() for every tree item

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#TREE_IAK1) LEVEL(1)
PARENT(#VIS_Tree) SOURCE(#VF_ELXAK1) VISIBLE(False)
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#TREE_IAK2) LEVEL(1)
PARENT(#VIS_Tree) SOURCE(#VF_ELXAK2) VISIBLE(False)
```

```
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#TREE_IAK3) LEVEL(1)
PARENT(#VIS_Tree) SOURCE(#VF_ELXAK3) VISIBLE(False)
DEFINE_COM CLASS(#PRIM_TVCL) NAME(#TVCL_BOT) LEVEL(2)
PARENT(#VIS_Tree) SOURCE(#VF_ELBOT) VISIBLE(False)
```

* Currently focused tree item

```
Define_com #Prim_Objt #FocusTreeItem Reference(*dynamic)
```

* UI Control Definitions

```
Define Field(#UI_ISDEAF) Reffld(#VF_ELBOOL)
Def_Cond Name(*UI_LISTEN) Cond('#UI_IsDeaf *ne TRUE')
```

* Tree node tracking - track department nodes and department section nodes

```
DEFINE_COM CLASS(#Prim_kcol<#Prim_tvit #deptment>)
NAME(#DepNodes) STYLE(Collection)
DEFINE_COM CLASS(#Prim_kcol<#Prim_tvit #std_texts>)
NAME(#DepSecNodes) STYLE(Collection)
```

*

=====

* Method Definitions

*

=====

* -----

* Redefine the standard uClearInstanceList method

* -----

```
MthRoutine uClearInstanceList Options(*Redefine)
```

```
Invoke #DepNodes.RemoveAll
```

```
Invoke #DepSecNodes.RemoveAll
```

```
Clr_List #Vis_Tree
```

```
Set_ref #FocusTreeItem *null
```

EndRoutine

```
* -----  
* Redefine the standard uAddListEntry method  
* -----
```

Mthroutine Name(uAddListEntry) Options(*Redefine)

Define_com #Prim_tvit #DepParent Reference(*Dynamic)

Define_com #Prim_tvit #SecParent Reference(*Dynamic)

* The filter(s) supply SECTIONS and EMPLOYEES business objects in the instance

* list, but this instance list browser has decided to create a 3 level tree to

* visualize the data slightly differently. Handle each business object type differently

CASE OF_FIELD(#BusinessObjectType.Value)

```
* =====  
* SECTIONS business object  
* =====
```

WHEN VALUE_IS(= SECTIONS)

* Check for a parent department node in the tree and add one if required

Set_ref #DepParent #DepNodes<#AKey1.Value>

If_ref #DepParent is(*null)

#VF_ELXAK1 := #AKey1.Value

#VF_ELXAK2 := "

#VF_ELXAK3 := "

#VF_ELBOT := "

#VF_ELXVI1 := #AKey1.Value

#VF_ELXVI2 := 'Department ' + #AKey1.Value

#VF_ELXCA1 := "

```

#VF_ELXCA2 := "
#POSTCODE := 0

Add_Entry #Vis_Tree

Set_Ref #DepParent #Vis_Tree.CurrentItem

Set_Ref #DepNodes<#AKey1.Value> #DepParent

Endif

* Now add in the section node as a child of the department and keep a track of
it

#VF_ELXAK1 := #AKey1.Value
#VF_ELXAK2 := #AKey2.Value
#VF_ELXAK3 := #AKey3.Value
#VF_ELBOT := #BusinessObjectType.Value
#VF_ELXVI1 := 'Section ' + #AKey2.Value
#VF_ELXVI2 := #AColumn1.Value
#VF_ELXCA1 := "
#VF_ELXCA2 := "
#POSTCODE := #Ncolumn1.Value

Add_Entry #Vis_Tree

Set #Vis_Tree.CurrentItem ParentItem(#DepParent)

Set_Ref #DepSecNodes<(#AKey1.Value + '!' + #Akey2.Value)>
#Vis_Tree.CurrentItem

* =====
* EMPLOYEES business object
* =====

WHEN VALUE_IS(= EMPLOYEES)

#VF_ELXAK1 := #AKey1.Value
#VF_ELXAK2 := #AKey2.Value

```

```

#VF_ELXAK3 := #AKey3.Value
#VF_ELBOT := #BusinessObjectType.Value
#VF_ELXVI1 := #VisualId1.Value
#VF_ELXVI2 := #VisualId2.Value
#VF_ELXCA1 := #AColumn1.Value
#VF_ELXCA2 := #AColumn2.Value
#POSTCODE := #NColumn1.Value

Add_Entry #Vis_Tree

* Final the section node (previously added) that will be this nodes tree parent

Set_Ref #SecParent #DepSecNodes<(#AKey1.Value + '.' + #Akey2.Value)>

* Show an error or set the parent item correctly ....

If_ref #SecParent is(*null)
Use message_box_show ( ok ok error *Component 'Attempt to add employee
with out a valid SECTIONS parent')
else
Set #Vis_Tree.CurrentItem ParentItem(#SecParent)
Endif

ENDCASE

* Finished

EndRoutine
* -----
* Determine whether to accept selection of new Sections from the grid
* -----
EVTROUTINE HANDLING(#Vis_Tree.ItemGotFocusAccept
#Vis_Tree.ItemGotSelectionAccept) Accept(#ACCEPT)
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
If (#vf_elbot *ne ' ')
If Cond(#avFrameworkManager.uCurrentLockStatus *EQ TRUE')
#ACCEPT := FALSE
#UI_ISDEAF := TRUE
Else

```

```

#ACCEPT := TRUE
#UI_ISDEAF := FALSE
Endif
Endif
ENDROUTINE
* -----
* Handle selection of an item from the tree
* -----
EvtRoutine Handling(#Vis_Tree.ItemGotSelection)
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)

If (#vf_elbot *ne ' ')

Signal SetSelectedInstance AKey1(#VF_ELXAK1) AKey2(#VF_ELXAK2)
AKey3(#VF_ELXAK1) BusinessObjectType(#VF_ELBOT)

* Handle the special case where the focus did not fire correctly

If_ref #FocusTreeItem is(*null)
Set_ref #FocusTreeItem #Vis_Tree.CurrentItem
Signal SetCurrentInstance AKey1(#VF_ELXAK1) AKey2(#VF_ELXAK2)
AKey3(#VF_ELXAK3) BusinessObjectType(#VF_ELBOT)
Endif

Endif
EndRoutine
* -----
* Handle unselection of an item from the tree
* -----
EvtRoutine Handling(#Vis_Tree.ItemLostSelection)
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
If (#vf_elbot *ne ' ')
Signal DropSelectedInstance AKey1(#VF_ELXAK1)
AKey2(#VF_ELXAK2) AKey3(#VF_ELXAK3)
BusinessObjectType(#VF_ELBOT)
Endif
EndRoutine
* -----
* Handle focus of an item from the tree

```

```
* -----  
EvtRoutine Handling(#Vis_Tree.ItemGotFocus)  
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)  
If Cond(*UI_LISTEN)  
If (#vf_elbot *ne ' ')  
Signal SetCurrentInstance AKey1(#VF_ELXAK1) AKey2(#VF_ELXAK2)  
AKey3(#VF_ELXAK3) BusinessObjectType(#VF_ELBOT)  
Endif  
Endif  
EndRoutine  
* -----  
* Handle loss of focus of an item from the tree  
* -----  
EvtRoutine Handling(#Vis_Tree.ItemLostFocus)  
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)  
If (#vf_elbot *ne ' ')  
Signal DropCurrentInstance AKey1(#VF_ELXAK1) AKey2(#VF_ELXAK2)  
AKey3(#VF_ELXAK3) BusinessObjectType(#VF_ELBOT)  
Endif  
EndRoutine  
  
End_Com
```

Advanced Instance List Processing

Creating a single shared 'manager' for your business objects allows you to centralize and standardize all your instance list activities to avoid repeating code in your filters or command handlers.

Using a shared manager you can:

- Create shared methods and make them available to all your filters and command handlers
- Delegate activities to the manager
- Manage the notification of events performed by filters and command handlers

For more information, see:

[Avoid Duplicated Instance List Code](#)

[Centralize all your Instance List Activities](#)

[Moving towards Real Business Object Management](#)

[Manipulate Instance Lists from RAMP Scripts](#)

[Delegate Common Tasks to your own Instance List 'Manager'](#)

[Low Level Direct Access to the Visualization Trees.](#)

Avoid Duplicated Instance List Code

If you have many different filters then you may end-up with duplicated instance list code. This may be annoying if a designer decides to change the key structure or additional columns that are used in an instance list, because you have to change all the filters to use the new instance list format.

To avoid this, do what you always do, and centralize the instance list manipulation code into one Visual LANSA reusable part that all your filters share.

Imagine a Visual LANSA reusable part named EMPMNGR that adds employees to an instance list in 3 different ways:

```
BEGIN_COM ROLE(*EXTENDS #PRIM_OBJT)

* Perform employee searches and add to the instance list

MthRoutine PerformSearch
Define_Map *Input #vf_lm002 #ListManager Pass(*By_Reference)
Define_Map *Input #std_num #SearchType
Define_Map *Input #EmpNo #UseEmpNo Mandatory(' ')
Define_Map *Input #SurName #UseSurName Mandatory(' ')
Define_Map *Input #PostCode #UsePostCode Mandatory(0)
Define_Map *input #Prim_Boln #Clear mandatory(true)

Invoke #ListManager.BeginListUpdate

If (#Clear)
Invoke #ListManager.ClearList
Endif

Case #SearchType

when (= 1)
Select fields(*all) from_file(pslmst) with_key(#UseEmpno) Generic(*Yes)
#Com_Owner.AddEmployeeToList ListManager(#ListManager)
Endselect

when (= 2)
Select fields(*all) from_file(pslmst2) with_key(#UseSurName) Generic(*Yes)
```

```

#Com_Owner.AddEmployeeToList ListManager(#ListManager)
Endselect

when (= 3)
Select fields(*all) from_file(pslmst) where(#PostCode = #UsePostCode)
#Com_Owner.AddEmployeeToList ListManager(#ListManager)
Endselect
EndCase

Invoke #ListManager.EndListUpdate

Endroutine

* Add an employee to the instance list

```

MthRoutine AddEmployeeToList Access(*Private)

```

Define_Map *Input #vf_lm002 #ListManager Pass(*By_Reference)

#FullName := #GiveName + " " + #SurName

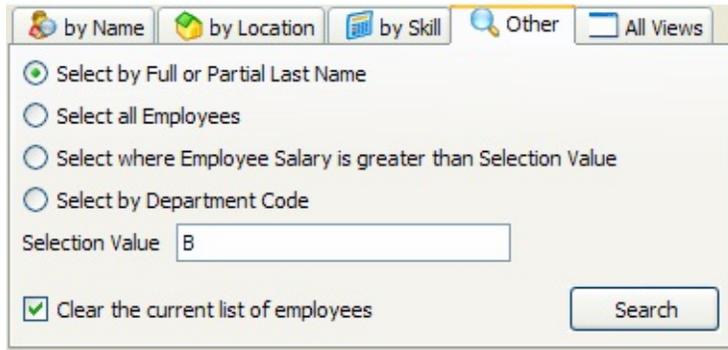
Invoke Method(#ListManager.AddtoList) Visualid1(#Empno)
Visualid2(#FullName) Akey1(#Deptment) Akey2(#Section) Akey3(#Empno)
AColumn1(#Phonehme) AColumn2(#Address1) nColumn1(#PostCode)

Endroutine

END_COM

```

Now imagine you have five employee filters like this:



If these filters each declared the shared VL reusable part like this:

```
DEFINE_COM CLASS(#EMPMNGR) NAME(#EmployeeManager)
scope(*Application)
```

then they can perform their respective searches by using a single command like this:

```
Invoke #EmployeeManager.PerformSearch ListManager(#avListManager)
SearchType(1) UseEmpNo(#EmpNo) Clear(true)
```

Or like this:

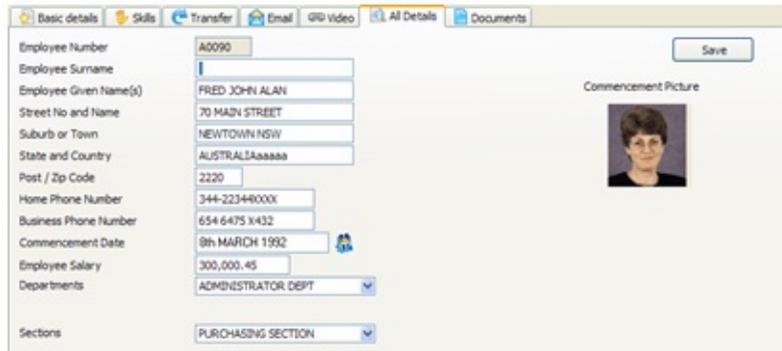
```
Invoke #EmployeeManager.PerformSearch ListManager(#avListManager)
SearchType(2) UseSurName(#SurName) Clear(false)
```

It is worth noting:

- That the employee manager #EMPMNGR object uses EXTENDS(#PRIM_OBJT). This defines it as a primitive object that has no visual context.
- The use of Scope(*Application) when declaring #EMPMNGR. This makes sure that only a single #EMPMNGR object is ever created and it is shared by all your filters or command handlers.
- That the filters might not have to have any search logic at all in them, nor would they have to directly access the database. All their search and database activities could be handled by the common shared #EMPMNGR object.
- If you need to change the employee instance list keys or the additional columns you now only need to change and recompile #EMPMNGR.

Centralize all your Instance List Activities

Once you start to centralize all your instance lists management code into a single manager component you will find it can be used in many different ways. For example, command handlers often need to update details in the instance list when the user updates something. For example, this command handler would need to update the instance list if the user changed the employees' name:



The screenshot shows a web-based form for managing employee details. The form is titled 'Basic details' and includes a 'Save' button. The fields are as follows:

Field	Value
Employee Number	A0090
Employee Surname	
Employee Given Name(s)	FRED JOHN ALAN
Street No and Name	70 MADN STREET
Suburb or Town	NEWTOWN NSW
State and Country	AUSTRALIAaaaaa
Post / Zip Code	2220
Home Phone Number	344-223440000
Business Phone Number	654 6475 X432
Commencement Date	08h MARCH 1992
Employee Salary	300,000.45
Departments	ADMINISTRATOR DEPT
Sections	PURCHASING SECTION

If #EMPMNGR had a shared method like this:

```
MthRoutine UpdateListDetails
Define_Map *Input #vf_lm002 #ListManager Pass(*By_Reference)
Define_Map *Input #EmpNo #ForEmpNo

Fetch fields(*all) from_file(PSLMST) with_key(#ForEmpno)

Invoke #ListManager.BeginListUpdate Mode(DYNAMIC)

#FullName := #GiveName + " " + #SurName

Invoke Method(#ListManager.UpdateListEntryData) Visualid1(#Empno)
Visualid2(#FullName) Akey1(#Deptment) Akey2(#Section) Akey3(#Empno)
AColumn1(#Phonehme) AColumn2(#Address1) nColumn1(#PostCode)

Invoke #ListManager.EndListUpdate

EndRoutine
```

Then all any command handler would need to do to update the instance list is

declare the shared #EMPMNGR like this ...

```
DEFINE_COM CLASS(#EMPMNGR) NAME(#EmployeeManager)
scope(*Application)
```

and then invoke the method to update the employees' instance list details

```
#EmployeeManager.UpdateListDetails ListManager(#avListManager)
ForEmpno(#Empno)
```

The benefit here of course is that all command handlers working with employee information could reuse the UpdateListDetails method and none of them even need to understand how this is done.

Sometimes command handlers also need to delete individual entries from the instance list, so if the centralized instance list manager contained a method like this:

```
MthRoutine DeleteListDetails
Define_Map *Input #vf_lm002 #ListManager Pass(*By_Reference)
Define_Map *Input #EmpNo #ForEmpNo
Define_Map *Input #Deptment #InDepartment
Define_Map *Input #Section #InSection

Invoke #ListManager.BeginListUpdate Mode(DYNAMIC)

Invoke Method(#ListManager.RemoveFromList) Akey1(#InDeptment)
Akey2(#InSection) Akey3(#ForEmpno)

Invoke #ListManager.EndListUpdate

EndRoutine
```

Then any command handler would only need to declare the shared manager:

```
DEFINE_COM CLASS(#EMPMNGR) NAME(#EmployeeManager)
scope(*Application)
```

and invoke the method like this:

```
#EmployeeManager.DeleteListDetails ListManager(#avListManager)
ForEmpno(#Empno) InDepartment(#Department) InSection(#Section)
```

to delete the details of an employee from an instance list.

Moving towards Real Business Object Management

If you extrapolate the ideas in the preceding section, you might start to think "Why couldn't the manager perform the actual data base updates and deletes as well?"

If the instance list manager #EMPMNGR has methods like UpdateListDetails and DeleteListDetails that work on the employees' in the instance list, why couldn't it have methods named UpdateEmployee and DeleteEmployee that updated or deleted the actual database rows in a centralized way, and then automatically reflected the change into the instance list as well?

Indeed it could, but that's getting into the area of true business objects managers, which is a bit beyond the scope of this document.

Manipulate Instance Lists from RAMP Scripts

Filters and command handlers can more easily manipulate the instance list by invoking simplified methods in a shared instance list 'manager'. To do this all they have to do is

- Declare the shared instance list manager using Scope(*Application)
- Invoke the methods in exposes.

RAMP 5250 navigation scripts cannot declare a VL reusable component and invoke its methods, but what they can do is signal events, so if 'something' is listening for the events they signal, then it can invoke the manager methods on their behalf.

Typically the 'something' that is listening for RAMP script events is a filter, which has an EVTROUTINE something like this in it:

```
Evtroutine Handling(#Com_owner.avEvent) WithId(#EventId)
    WithAInfo1(#AInfo1) WithAInfo2(#AInfo2) WithAInfo3(#AInfo3)
    Options(*NOCLEARMESSAGES *NOCLEARERRORS)

Case #EventId.Value

When (= UPDATE_EMPLOYEE_5250)

#EmployeeManager.UpdateListDetails ListManager(#avListManager)
ForEmpno(#AInfo1)

When (= DELETE_EMPLOYEE_5250)

#EmployeeManager.DeleteListDetails ListManager(#avListManager)
ForEmpno(#AInfo1) inDepartment(#AInfo2) InSection(#AInfo3)

EndCase

Endroutine
```

A filter with this code in it is listening for events named UPDATE_EMPLOYEE_5250 and DELETE_EMPLOYEE_5250. When it receives one of them it routes the request into the shared instance list 'manager'.

Of course there needs to be an agreed protocol between the RAMP script and the filter regarding how the employee key details are passed in the event payload parameters.

When a RAMP script detects a 5250 activity that means the instance list needs to be updated (eg: deleting or updating an employee) it would execute JavaScript script coded something like this:

```
AVSIGNALEVENT("DELETE_EMPLOYEE_5250", "BUSINESSOBJECT",  
objListManager.AKey1[0], objListManager.AKey2[0],  
objListManager.AKey3[0] );
```

Or

```
AVSIGNALEVENT("UPDATE_EMPLOYEE_5250",  
"BUSINESSOBJECT", objListManager.AKey3[0]);
```

Delegate Common Tasks to your own Instance List 'Manager'

So far we have seen how the task of adding to, updating and deleting instance list entries may be delegated to a common shared instance list 'manager'.

Any other activity that is likely to involve repeated code in a filter or a command handler can usually be delegated in the same way.

For example, in the preceding section RAMP scripts can access the instance list 'manager' by signaling an event to it via a listening filter. Since you will not always know which filter is listening, you would have to repeat the listening code in every filter.

One way to delegate this listening task, and virtually any listening task, to a common shared instance list manager is as follows:

In the instance list 'manager' define 2 collections like this:

```
* Keep track of all registered active filters and command handlers
```

```
DEFINE_COM CLASS(#Prim_Acol<#VF_ac007>) NAME(#ActiveFilters)
DEFINE_COM CLASS(#Prim_Acol<#VF_ac010>)
NAME(#ActiveHandlers)
```

Then add 2 methods like this:

```
MthRoutine RegisterInitialize
Define_map *input #VF_AC007 #Filter Pass(*By_Reference)
Mandatory(null)
Define_map *input #VF_AC010 #Handler Pass(*By_Reference)
Mandatory(null)
```

```
* Keep track of registered and active filters
```

```
If_ref #Filter is_not(*null)
Invoke #ActiveFilters.Insert Item(#Filter)
Endif
```

```
* Keep track of all registered and active command handlers
```

```
If_ref #Handler is_not(*null)
Invoke #ActiveHandlers.Insert Item(#Handler)
Endif
```

```
Endroutine
```

```
MthRoutine RegisterTerminate
```

```
Define_map *input #VF_AC007 #Filter Pass(*By_Reference)
```

```
Mandatory(null)
```

```
Define_map *input #VF_AC010 #Handler Pass(*By_Reference)
```

```
Mandatory(null)
```

```
* Remove the specified filter from the active collection
```

```
If_ref #Filter is_not(*null)
```

```
Invoke #ActiveFilters.Remove Object(#Filter)
```

```
Endif
```

```
* Remove the specified command handler from the active collection
```

```
If_ref #Handler is_not(*null)
```

```
Invoke #ActiveHandlers.Remove Object(#Handler)
```

```
Endif
```

```
Endroutine
```

Now, in each filter or command handler, you need to do three things:

- Define the instance list manager:

```
DEFINE_COM CLASS(#EMPMNGR) NAME(#EmployeeManager)  
scope(*Application)
```

- Register with the instance list manager when starting up. In a filter do this:

```
MthRoutine uInitialize Options(*Redefine)
Invoke #EmployeeManager.RegisterInitialize Filter(#Com_Owner)
Endroutine
```

In a command handler do this:

```
MthRoutine uInitialize Options(*Redefine)
Invoke #EmployeeManager.RegisterInitialize Handler(#Com_Owner)
Endroutine
```

(De)Register with the instance list manager when terminating. In a filter do this:

```
MthRoutine uTerminate Options(*Redefine)
Invoke #EmployeeManager.RegisterTerminate Filter(#Com_Owner)
Endroutine
```

In a command handler do this:

```
MthRoutine uTerminate Options(*Redefine)
Invoke #EmployeeManager.RegisterTerminate Handler(#Com_Owner)
Endroutine
```

Your instance list 'manager' is now aware all active filters and command handlers that are actively doing something with the business object (eg: Employees).

It can start to perform common shared actions on their behalf.

For example, if your instance list 'manager' had an event routine like this in it:

```

EvtRoutine Handling(#ActiveFilters<>.avEvent) WithId(#EventId)
WithAInfo1(#AInfo1) WithAInfo2(#AInfo2) WithAInfo3(#AInfo3)
COM_Sender(#SendingFilter) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)

Case #EventId.Value

When (= UPDATE_EMPLOYEE_5250)

#Com_Owner.UpdateListDetails ListManager(#SendingFilter.avListManager)
ForEmpno(#AInfo1)

When (= DELETE_EMPLOYEE_5250)

#Com_Owner.DeleteListDetails ListManager(#SendingFilter.avListManager)
ForEmpno(#AInfo1) inDepartment(#AInfo2) InSection(#AInfo3)

EndCase

Endroutine

```

then it has taken over the job of listening for RAMP script events for all filters. There is no longer any need to add this listening code to any specific filter.

The use of #ActiveFilters<>.avEvent is special. This instance list manager event routine is listening in to (or, eavesdropping, if you like) to all the filters that have registered with it. You can do the same sort of thing using #ActiveHandlers<> of course.

On the other side of this coin, an instance list manager can also signal events to all active filters and command handlers. Typically this is done like this:

First, define the events and methods to signal them into your instance lists 'manager', like in this example:

```

Define_Evt EmployeeUpdate
Define_map *input #Empno #EmployeeNumber

Define_Evt EmployeeDelete
Define_map *input #Empno #EmployeeNumber

```

```
Mthroutine Sig_EmployeeUpdate
Define_map *input #Empno #EmployeeNumber
Signal EmployeeUpdate EmployeeNumber(#EmployeeNumber)
Endroutine
```

```
Mthroutine Sig_EmployeeDelete
Define_map *input #Empno #EmployeeNumber
Signal EmployeeDelete EmployeeNumber(#EmployeeNumber)
Endroutine
```

Now, in any filter or command handler that wants to be notified when these events happen, you just need to put an event handling routine in to listen like this:

```
Evtoutine Handling(#EmployeeManager.EmployeeUpdate)
EmployeeNumber(#EmployeeNumber) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
```

```
Use Message_box_show (ok ok info *Component ("I have just been notified
that employee number " + #EmployeeNumber + " has been updated"))
```

```
Endroutine
```

```
Evtoutine Handling(#EmployeeManager.EmployeeDelete)
EmployeeNumber(#EmployeeNumber) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
```

```
Use Message_box_show (ok ok info *Component ("I have just been notified
that employee number " + #EmployeeNumber + " has been deleted"))
```

```
Endroutine
```

Finally, in the filter or command handler that wants to signal (or trigger, or notify) the event to others, you just need to do this:

```
#EmployeeManager.Sig_EmployeeUpdate EmployeeNumber(#Empno)
```

or

#EmployeeManager.Sig_EmployeeDelete EmployeeNumber(#Empno)

Low Level Direct Access to the Visualization Trees

In VLF-WIN code you can directly access the VL trees (class #PRIM_TRVW) that are used to visualize the instance list content.

There may be two trees – the primary tree and possibly a secondary tree when side-by-side or over-under displays are used.

Once you have a reference to the tree you can access the items within the tree and the columns within the items.

The properties #avListManager.avPrimaryTree and #avListManager.avSecondaryTree yield references to the primary and secondary VL trees used to visualize an instance list.

Some significant usage points:

- The trees are the visualizations of an instance list. They are not the instance list itself. If you change the way that an instance list has been visualized, you are not changing the instance list.
- Do not save tree or tree item references in your code. This would probably cause resource leakage or other problems.
- Do not change tree or tree items selection or focus. Update the instance list instead.
- Do not change tree or tree items inside beginlistupdate/endlistupdate code blocks.
- The tree references are not applicable if you are using your own snap-in instance list browser.
- Use caution with your changes and test carefully how they interact with the rest of the Framework.
- VL tree level programming skills are assumed.

Some examples:

Get the number of entries in primary visualization tree

```
#Std_text := #AVLISTMANAGER.avPrimaryTree.Entries.AsString  
Use MESSAGE_BOX_SHOW With_Args(ok ok info *COMPONENT ('Tree  
has ' + #Std_text + ' entries'))
```

Change the icon associated with every entry at level 1 in a tree

```
For Each(#TreeItem) In(#avListManager.avPrimaryTree.Items)  
#TreeItem.Image <= #vf_ic004
```

```
Endfor
```

Note: This example is iterating over the tree so it is working in the visual order of the currently displayed tree - not necessarily the order of the instance list entries.

Make sure the first entry in the tree is visible

```
#avListManager.avPrimaryTree.Items<1>.EnsureVisible := True
```

Note: This example does not cater for an empty tree.

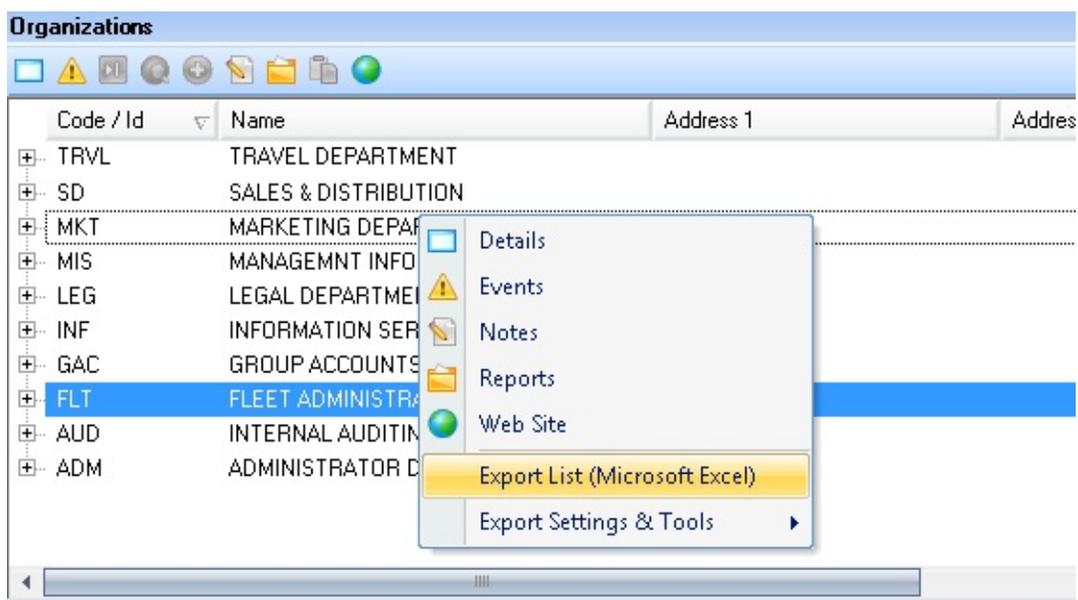
Export Instance List Contents

Applies to **Windows** and **.NET** applications

This feature allows end-user to easily export the entries in the instance list to a document. There is no additional effort required from the part of Framework developers. The supported document formats in VLF.NET are:

- Excel
- PDF
- HTML
- CSV

In the Windows environment, templates are not available and the only format currently supported is CSV.



When generating a Microsoft Excel report, the instance list tree structure can be retained through the use of the Outlining feature of Microsoft Excel.

Only entries that are available at the time of the report generation (which means that if you use a relationship handler to do on-demand loading of sub-rows, the report will include only entries that have been retrieved – which will happen when their parent node is expanded). Retrieved entries that are hidden because their parent node is collapsed will still be included in the report.

The entries will be exported in the order they appear at the time of the report generation. The same applies to the columns order (which means that if the end-user moves the column around, that's the setting that will be used in the report

generation).

- Getting Started
- Creating Your First Report Template
- Publishing Shared Templates
- Modifying Report Templates

Template Support

End-users or Framework developers can define their own report templates. Report templates are Excel documents that contain placeholders for actual report field values. Those placeholders will be substituted with real values during report generation. Think mail merge in Microsoft Word. Placeholders can easily be identified as they always start with “<#” and end with “>” (e.g. “<#dem_org.n1>”).

Note that the use of Excel document as templates does not limit the report output format to Excel only. Users can still generate Excel, PDF, and HTML reports. CSV format however does not support the use of templates (templates will be ignored when CSV format is selected).

To assist Framework developers and end-users in creating report templates, VLF.NET is capable of producing initial templates containing all the fields (i.e. the placeholders) from relevant business objects. Framework developers or end-users can then modify this template easily (remove unwanted columns, apply formatting etc).

Benefits of Using Excel Document as Templates

The use of Excel document as a report template allows the end-users to generate reports sophisticated enough to satisfy their needs. Any formatting applied to an Excel template will be fully applied to the reports generated based on that template.

Due to the simplicity of Microsoft Excel and its popularity amongst office workers, this feature will enable end-users to easily modify or even create their own templates, apply various formatting to suit their needs. Such a thing is unthinkable for more complex tools such as Crystal Reports.

Shared and Private Templates

There are two types of templates:

- **Shared** templates
- **Private** templates

Shared templates are basically public templates, shared across the company. Private templates are “My Templates” - they are accessible only to the users creating the templates. Private templates are stored in the “My Documents” folder of the user creating the private reports.

Shared templates should be packaged and then uploaded to your Framework

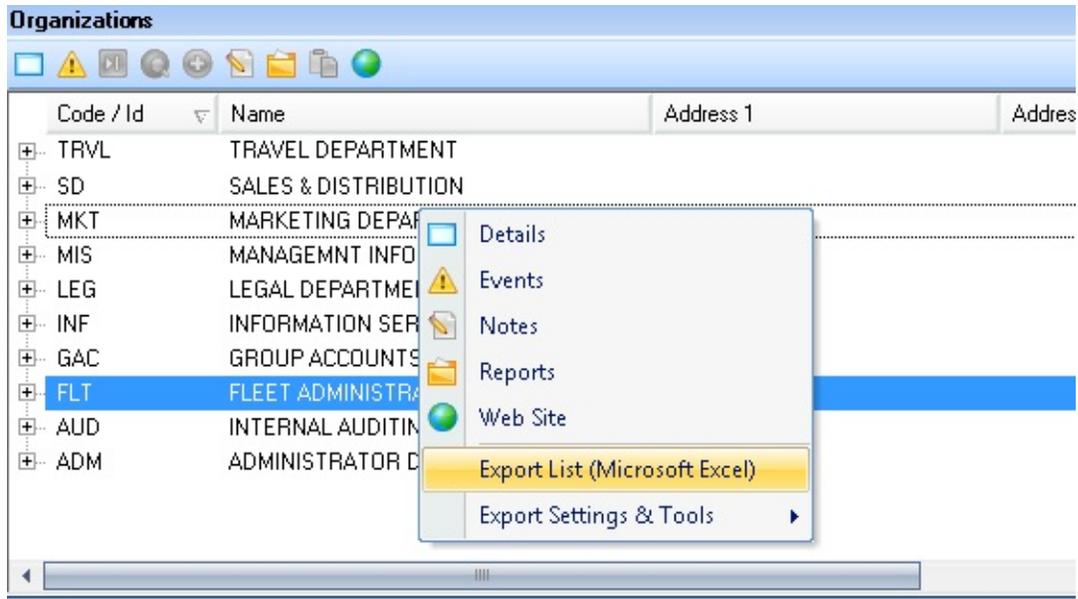
website.

Shipped Sample Templates

Sample templates are installed to the partition execute folder in a file called SampleReportTemplates.zip. You need to unzip the contents of the zip file to the “My Documents” folder on your PC in order to use them.

Getting Started

The reporting feature appears on the instance list context menu.



Notice the two new menu item at the bottom of the context menu:

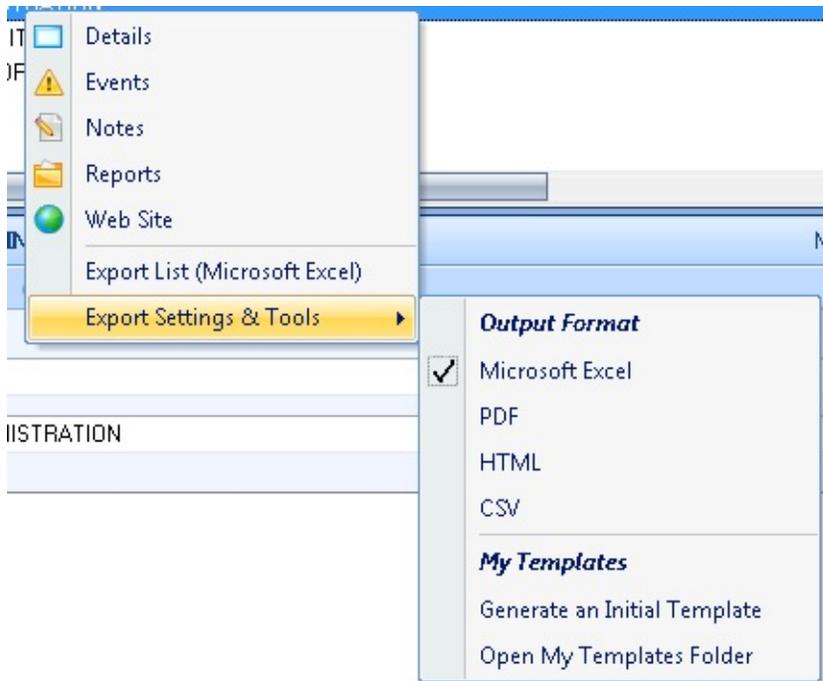
Export List (Microsoft Excel)

Generate a report. There is currently no report template available (and arrow will be displayed next to this menu item if there are templates available)

The current output format is Microsoft Excel document as indicated in the menu item.

Export Settings & Tools

Allows end-users to choose output format & manage their templates.



A click on the “Export List (Microsoft Excel)” will export the instance list entries to an Excel document without using any template (default formatting will be applied).

Microsoft Excel - Spreadsheet_20090127_033323_Jan.xls

File Edit View Insert Format Tools Data Window Help Type a question for help

Arial 10 B I U

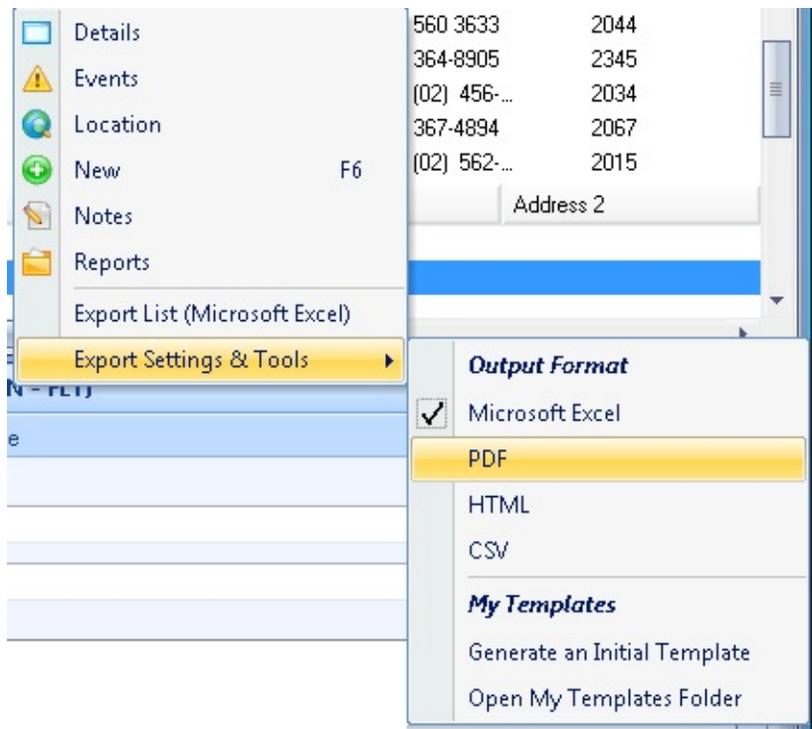
A1 Code / Id

	A	B	C
1	Code / Id	Name	Address 1
2	ADM	ADMINISTRATOR DEPT	
3	INTERNAL ADMIN SRV	01	125 Main Street 2
4	JONES2,BEN	A1001	144 Frog
5	PAUL,PATRICK	A1012	6 Camillo Avenue
6	PATTISON,GEORGE	A1013	12 Augusta Avenue
7	WOODS,BRADLEY	A1015	59 Darley Road,
8	DOUGLAS,ADAM PETER	A1020	6 Reading Avenue,
9	MCCULLY,DAVID	A1021	15 Baker Place,
10	ROBINSON,MARY	A1025	14 Whitby Road,
11	MORRISON,ALAN	A1027	47 Lincoln Street,
12	VEREY,WARREN PETER	A1111	1 Main Rd
13	MRS BRICK,GILL	A1404	22 Moton Street
14	REDFORD,ROBERT	A1509	122 Arthur Street
15	SURNAME201,GIVENAME1	A9006	A
16	SURNAME1,GIVENAME1	A9081	A
17	PURCHASING SECTION 2	02	123 Pacific Highwa
18	ACCOUNTING SECTION	03	252 Canterbury Ro:
19	SALES & MARKETING	04	121 Pitt Town Roac
20	MAINTENANCE	05	120 Railway Paradi
21	PERSONNEL SECTION	06	121 Smith St
22	VEHICLE MAINTENANCE	09	121 Smith Street
23	AUD	INTERNAL AUDITING	
24	FLT	FLEET ADMINISTRATION	

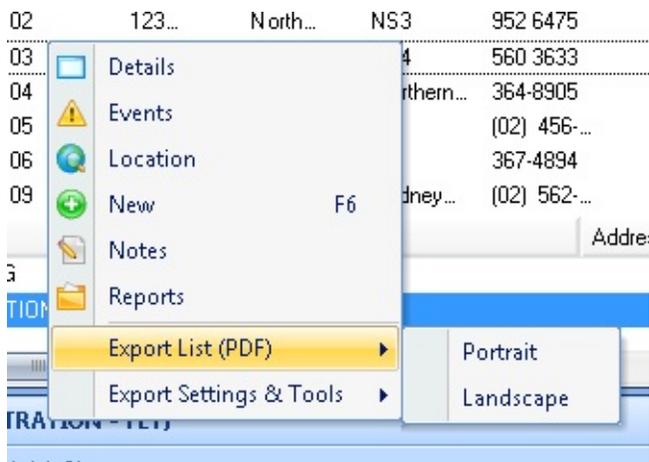
Sheet1

Ready

Now change the output format to PDF:



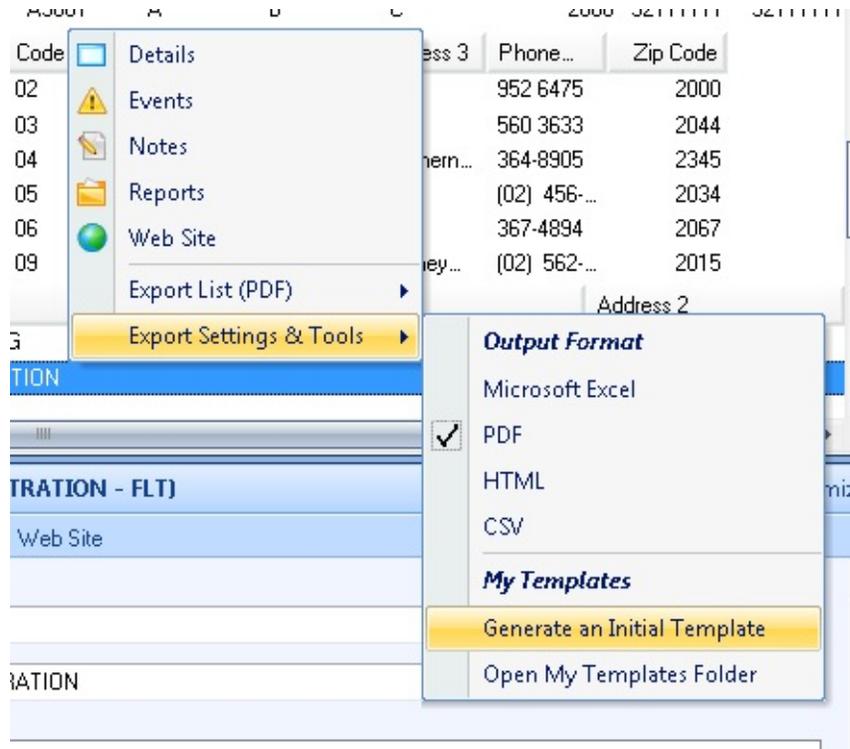
Reopen the context menu:



Notice that the Export List menu item now indicates that the current output format is PDF, and it can either generate **Portrait** or **Landscape** pages in the resulting PDF document.

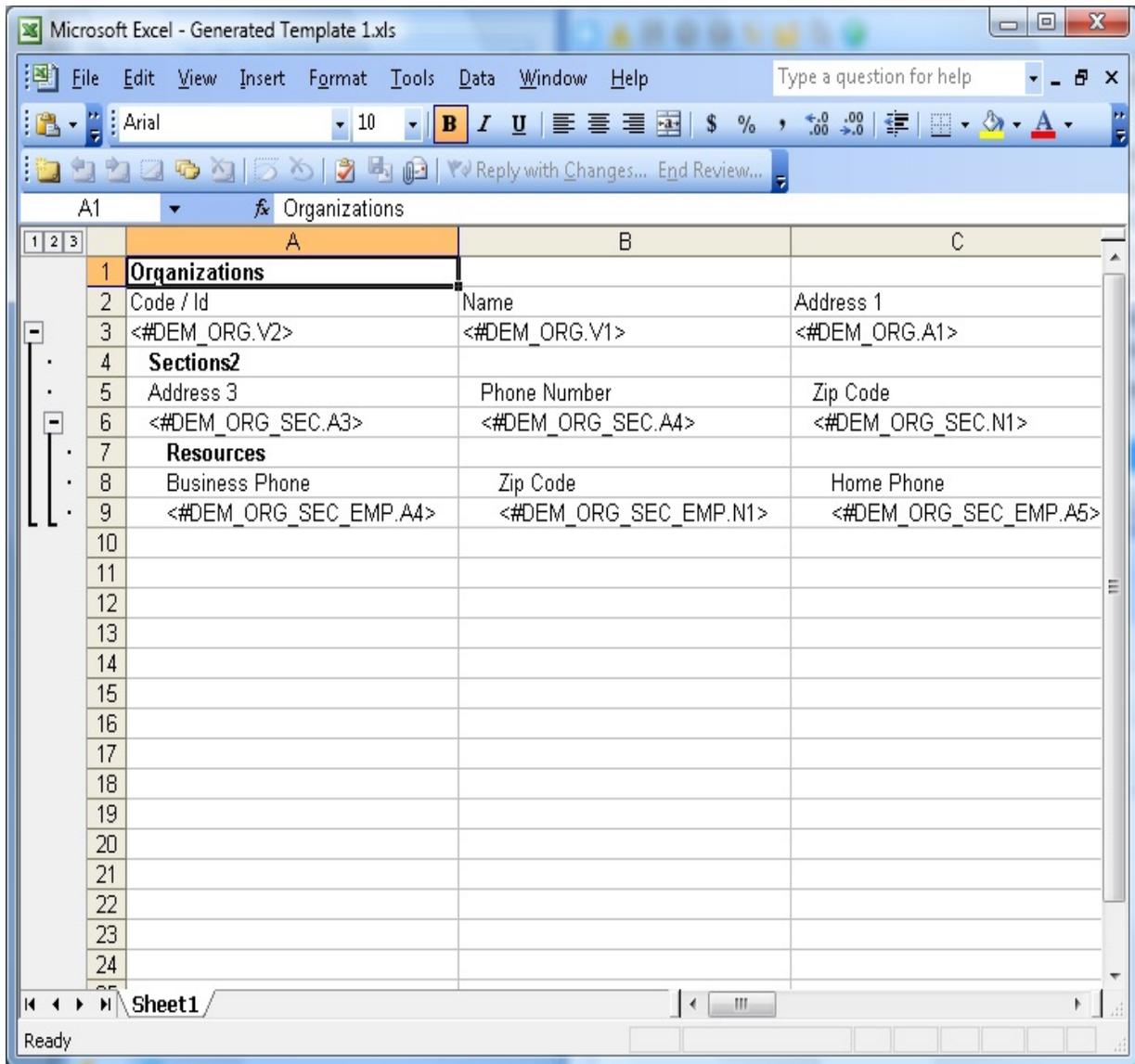
Creating Your First Report Template

1. Open the “Export Settings & Tools” menu



2. Click on “Generate an Initial Template”.

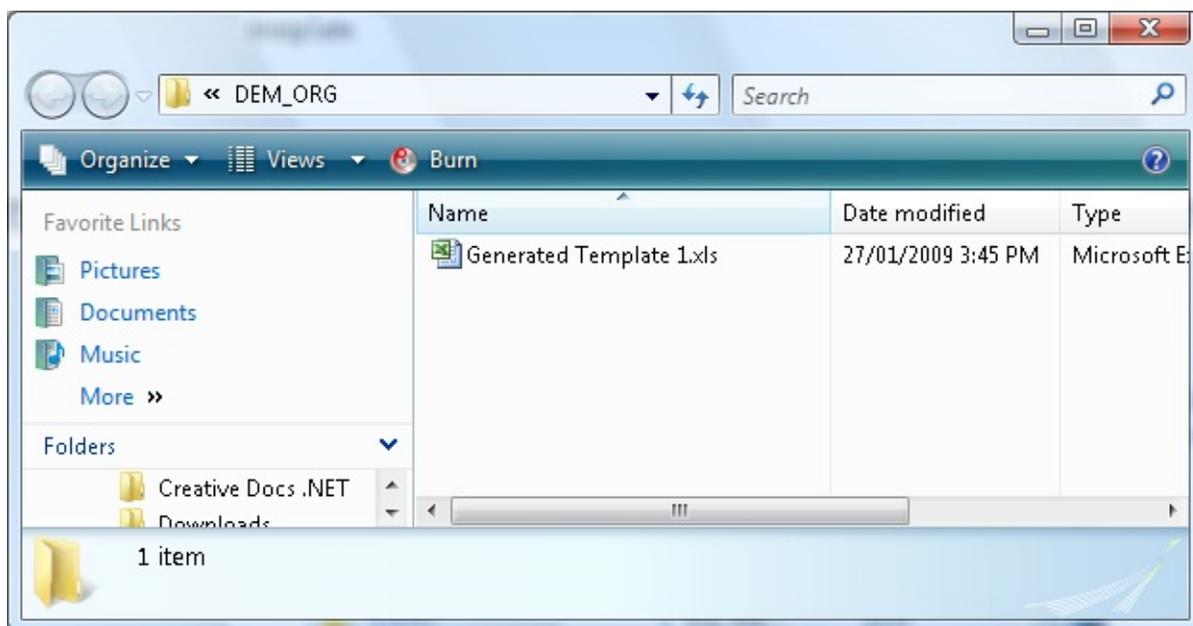
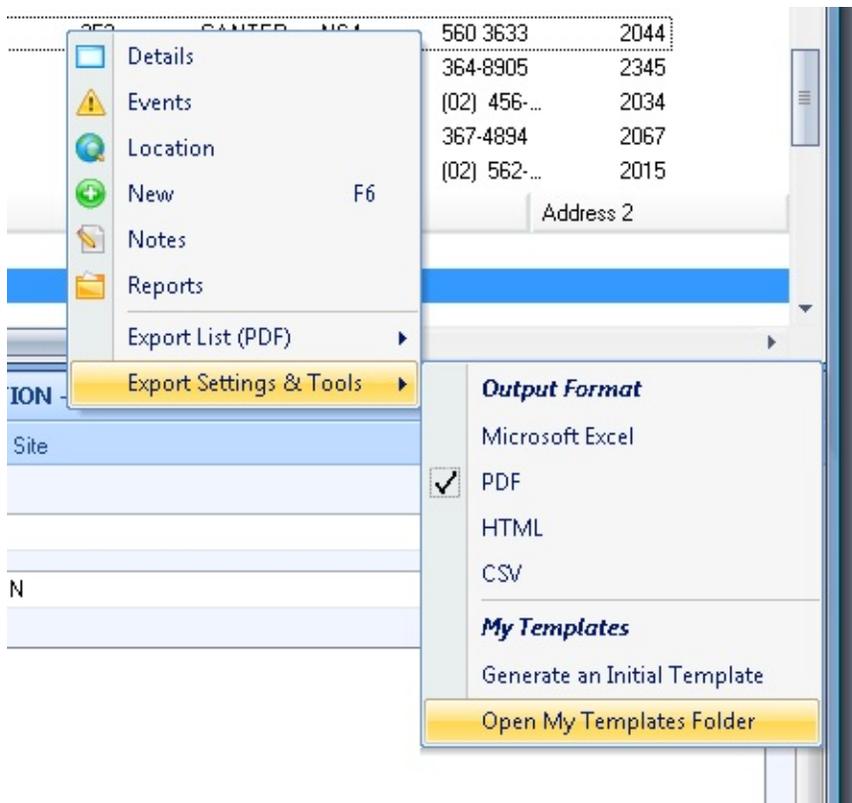
If Excel is installed it will open up showing the new template.



Pay attention to the “value placeholders” – those texts in the form of <#BusinessObject.Field>.

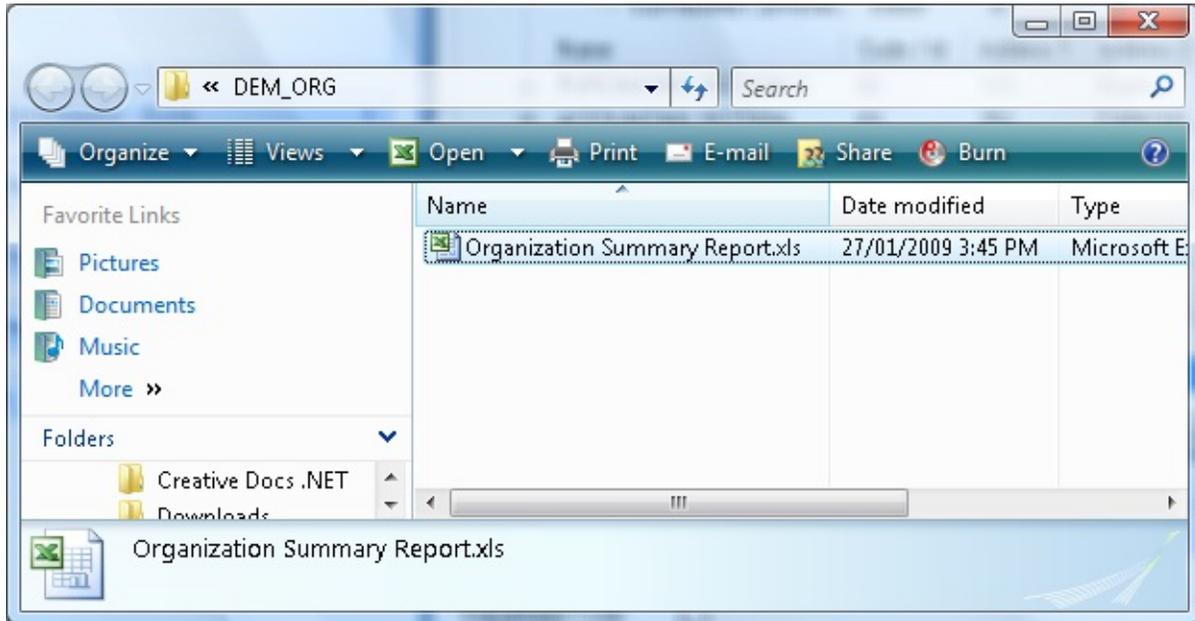
3. Save your template, close it, and rename it to a more descriptive name.

As mentioned earlier, report templates are stored under “My Documents” folder. To rename your report template, you will need to open the folder containing the template. You can easily do that from the context menu.

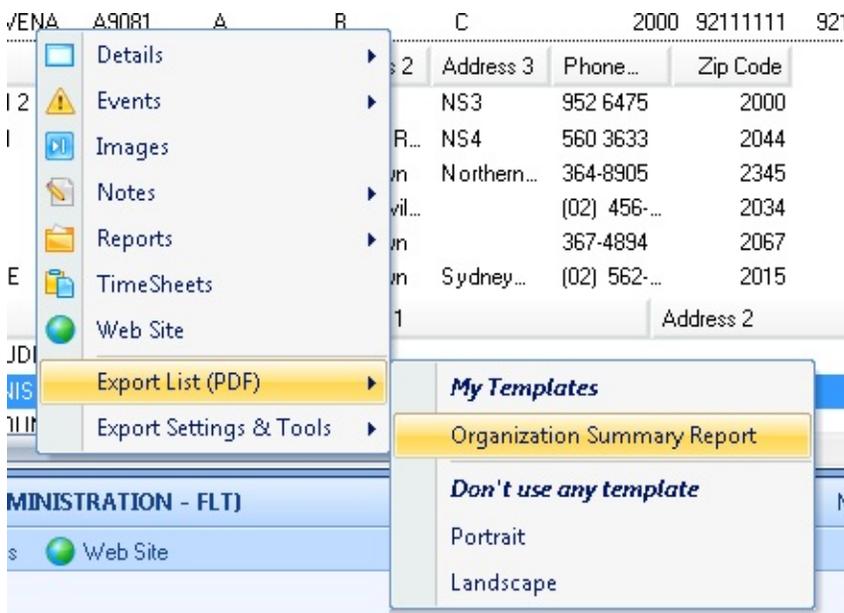


4. You can now rename the template file, say to “Organization Summary

Report”.



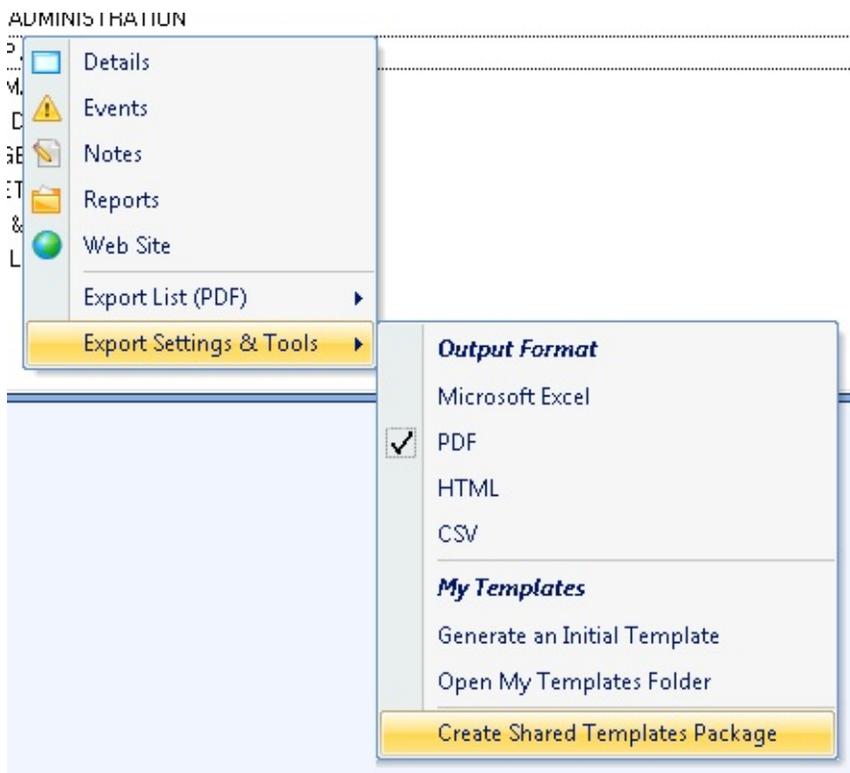
When you go back to the context menu, your report template will now appear under “My Templates”



Publishing Shared Templates

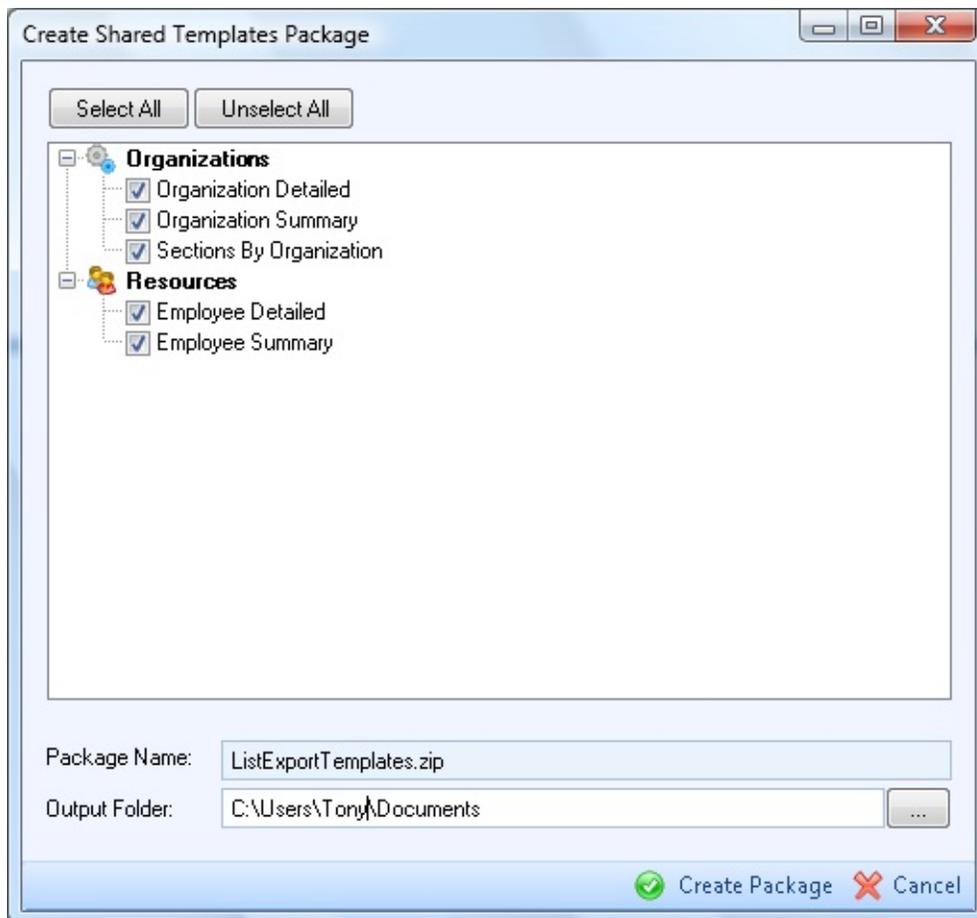
To share your templates with other people, you will need to create a template package (which is basically a ZIP file containing the templates you want to share), and then get your Framework web administrator to upload the package to the Framework website.

1. Open the context menu, and then select **Create Shared Templates Package**.

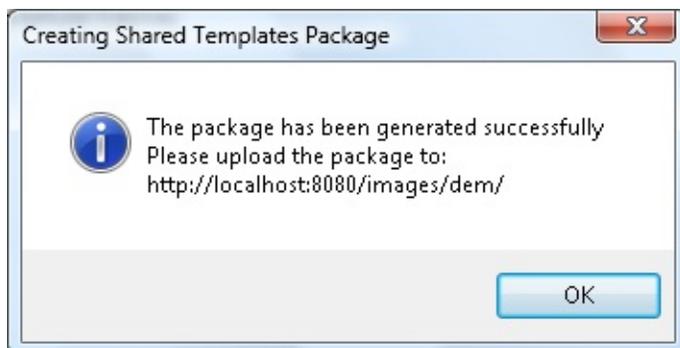


A dialog box will appear, showing you all your private templates.

2. Select the templates you want to share and the output folder under which the package should be created. Click on **Create Package**.



If everything is OK, a message similar to below will appear, and then a new Windows Explorer window will open showing the output folder. Upload the **ListExportTemplates.zip** file to your Framework website (ask your network/web administrator to do that if necessary).



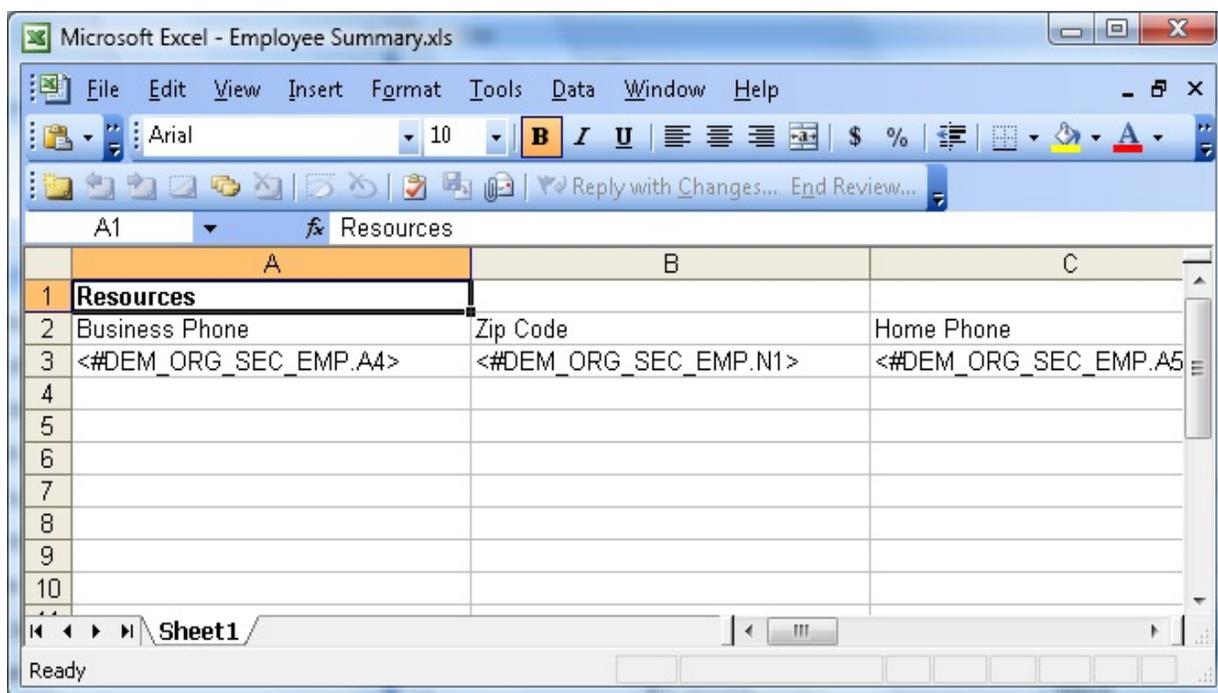
Modifying Report Templates

You can modify the initial report templates in any way you like (just be careful not to change the field names), however there is one thing that you that you need to be aware of before you start changing the templates: named cell range. Named cell range is an Excel's feature that allows a range of cell to be given a descriptive name. VLF.NET makes use of this particular feature to identify which part of the Excel worksheet should be repeated for each record (when you have a tabular report, you wouldn't want to repeat the report title and column headers for each record).

Let's use the **Resources** business object from the demo framework to illustrate this concept.

The following explanation is based on Office Excel 2003.

1. Get VLF.NET to create a new template for you.

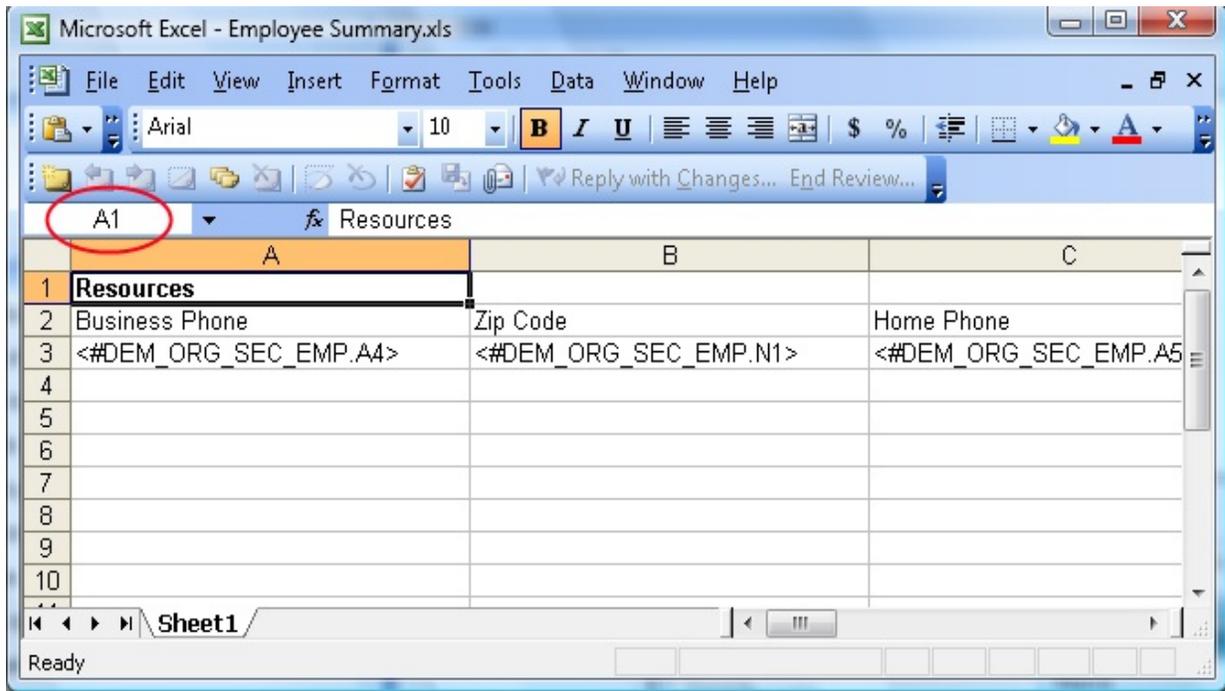


It's quite clear that we would want row number 3 to be repeated for each record, but not row number 1 and 2.

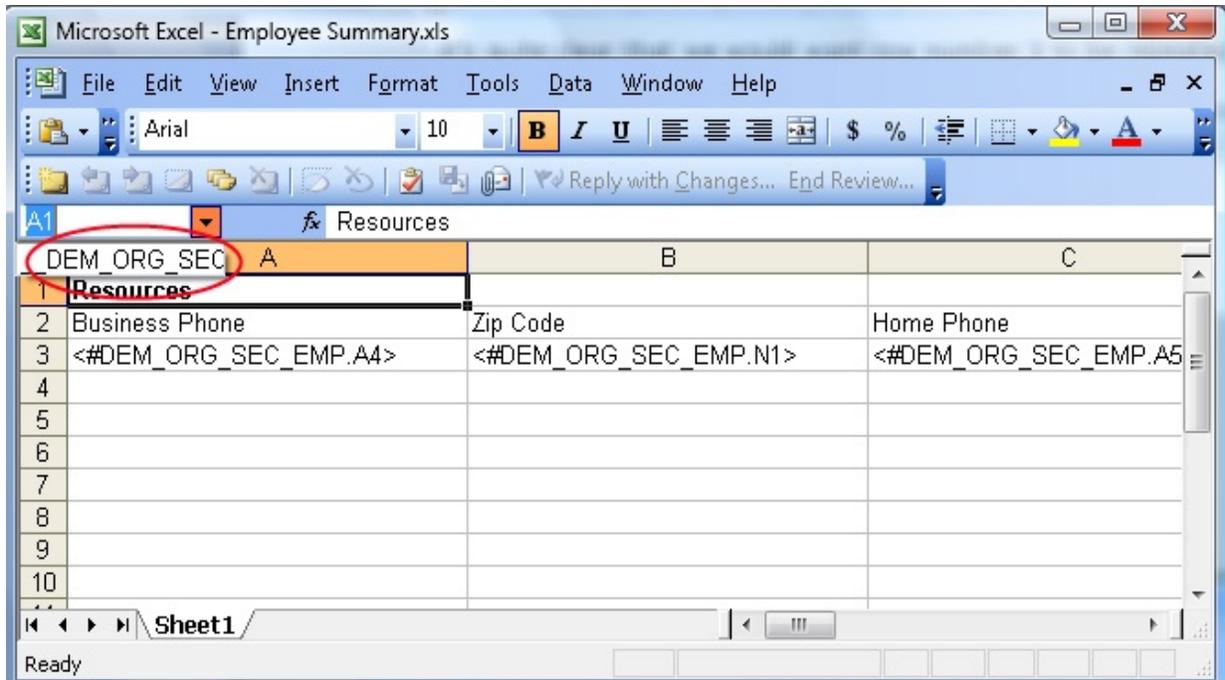
When this template is used in generating the actual report, VLF.NET understands that you want only to repeat row 3 because the a special name has

been assigned to row 3.

1. To see that, click the area marked in the screenshot below:

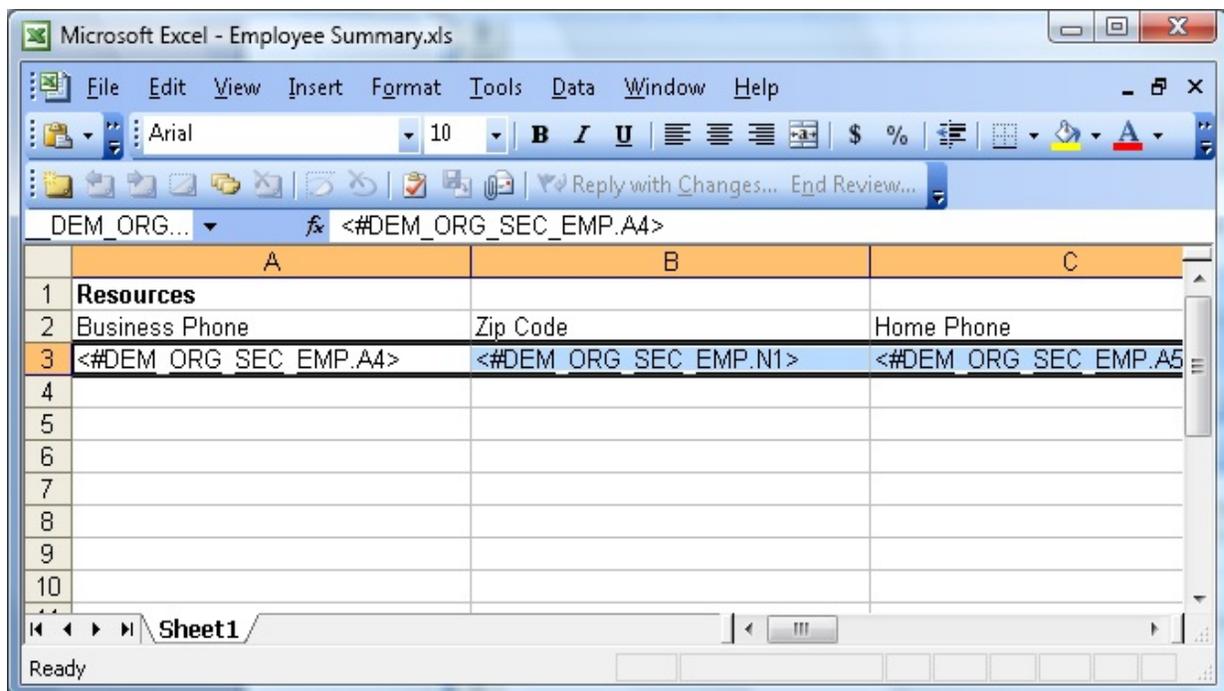


After you click it, it will drop down and display the named range defined for this worksheet.



This template contains only one named range which is `__DEM_ORG_SEC_EMP__` (you can only see `__DEM_ORG_SEC` as it's truncated). The number of named range defined in a sheet corresponds to the number of business objects used in the template.

If you click on `__DEM_ORG_SEC_EMP__` in that dropdown, Excel will show to you the cell range associated with that name.

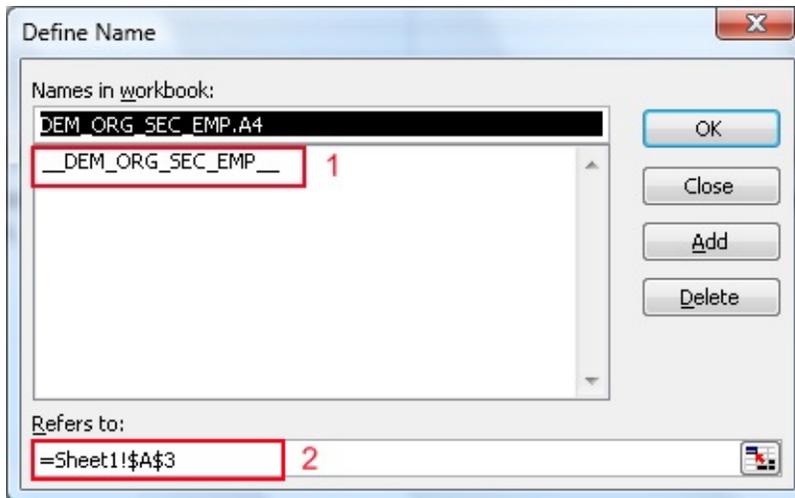


It's important to you to understand this concept when you start modifying your templates. There will be times when you would need to adjust the named range when you insert new rows.

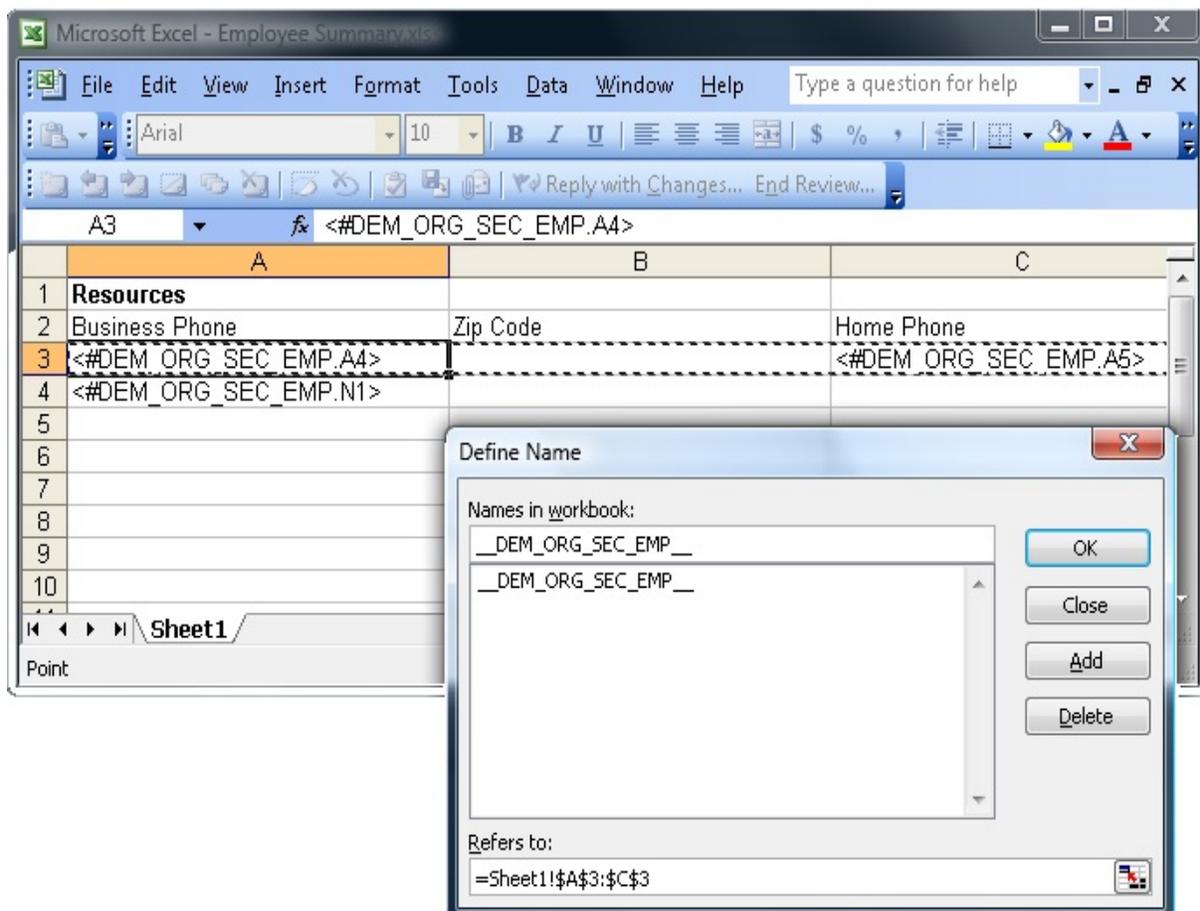
Let's say that you want to split the one row into 2 rows (so that each record will generate 2 rows). The second row will contain the Zip Code only. To do this, move `<#DEM_ORG_SEC_EMP.N1>` to cell A4.

You now need to extend the defined range to cover row 4 as well.

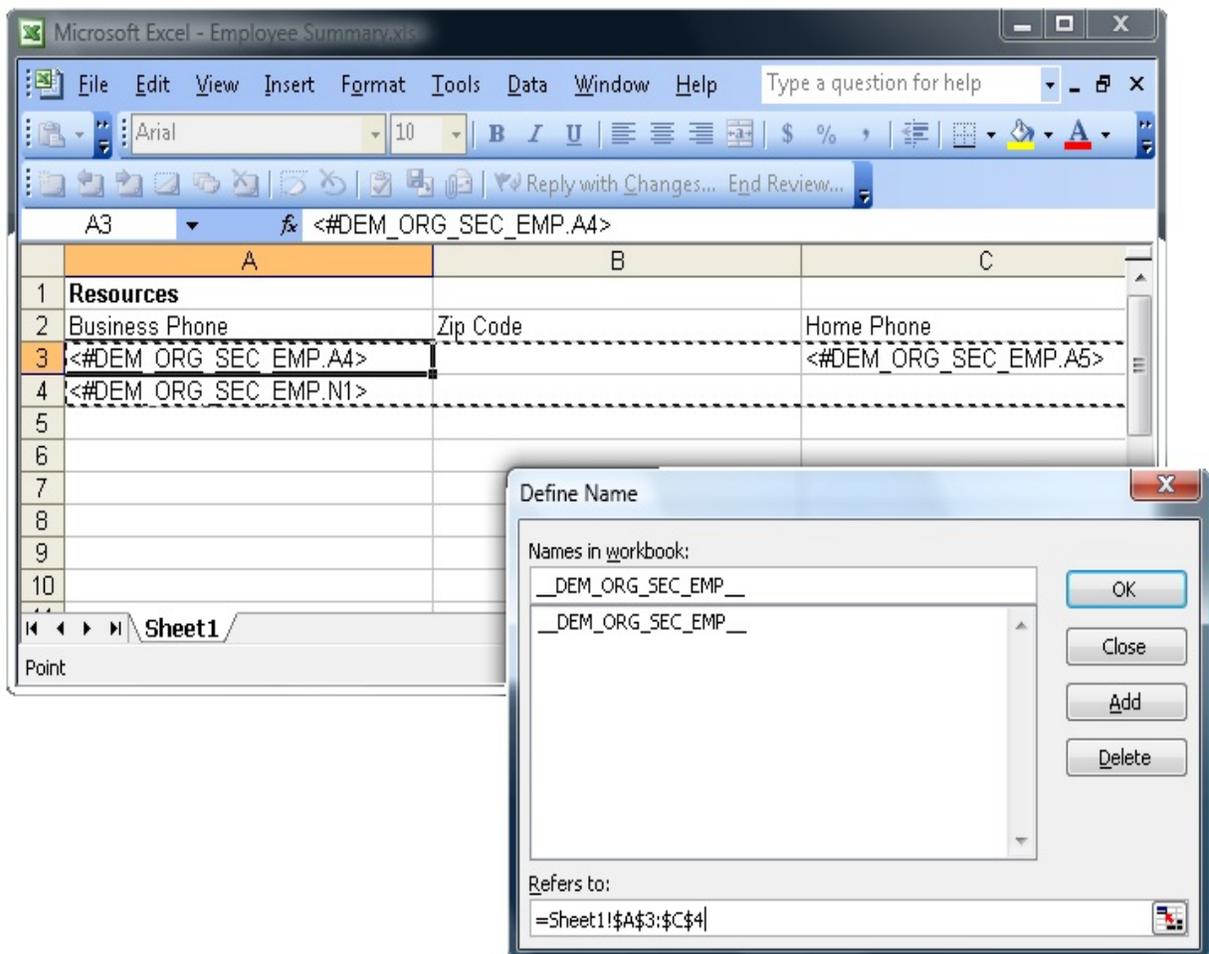
2. To do this, open the **Insert** menu, and then choose **Name**, and then **Define**.



3. Click on the item on the list, and then click on the range text box. Excel will indicate the current range associated with `___DEM_ORG_SEC_EMP___`.



4. Extend the range either by changing row 3 to 4 or by pressing **shift** and then **arrow down** key. Make sure you press the **OK** button afterwards.



VLF.NET Excel Report Template Design Frequently Asked Questions

Can I have a running total figure in my report?

Can I include charts in my report?

Is there a way to configure a template to force all data to be retrieved first prior to generating the report?

How can I put printing date & time on the report?

How can I insert the username of the user who generates the report?

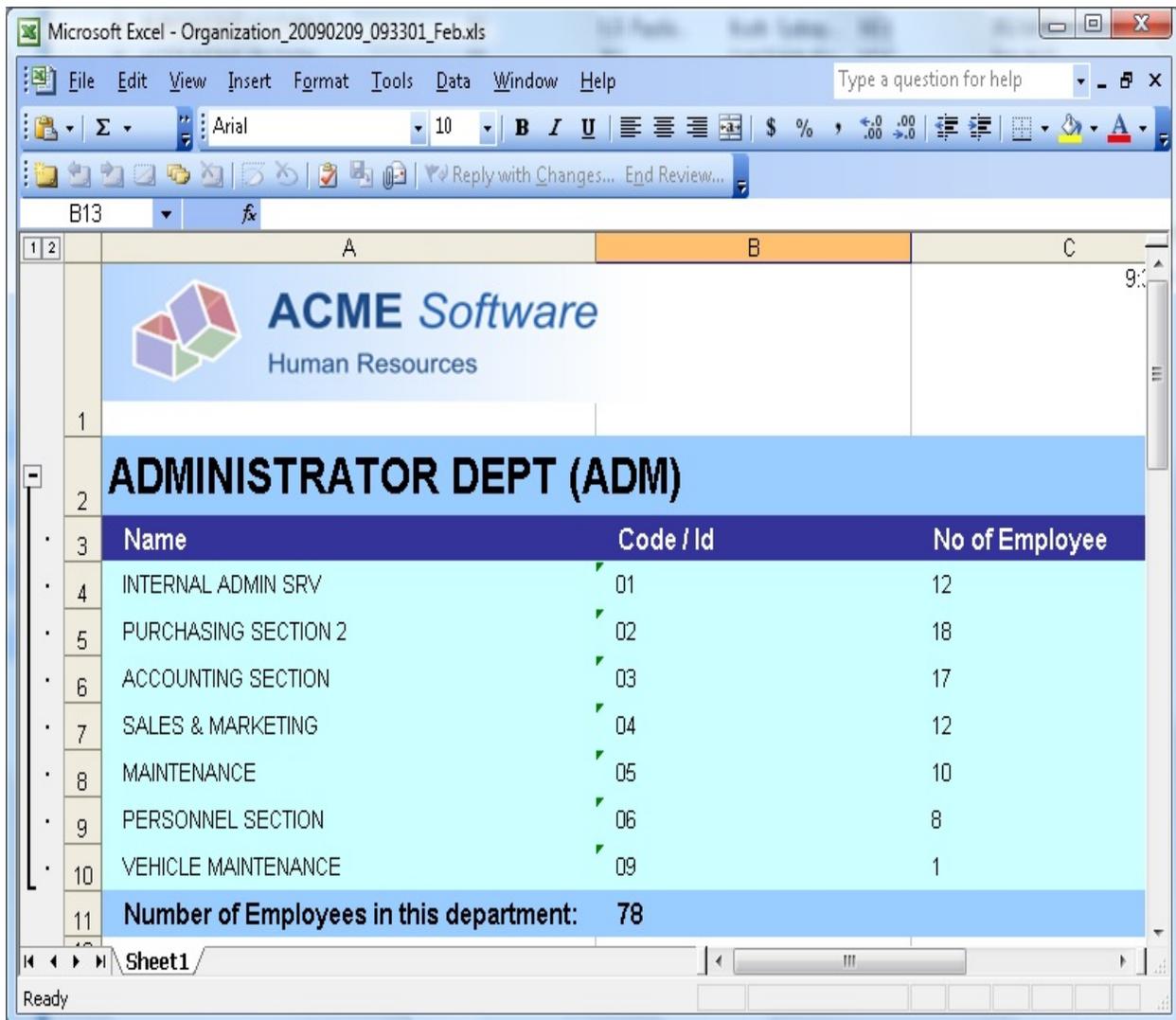
How do I keep some rows together in a page?

How can I tell a template to fit all columns in a page when generating reports?

Can I have a running total figure in my report?

Yes. Let's say that you have a report of sections grouped by department. You have the number of employees for each section, but not the number of employees per department, so it will have to be calculated as a running total.

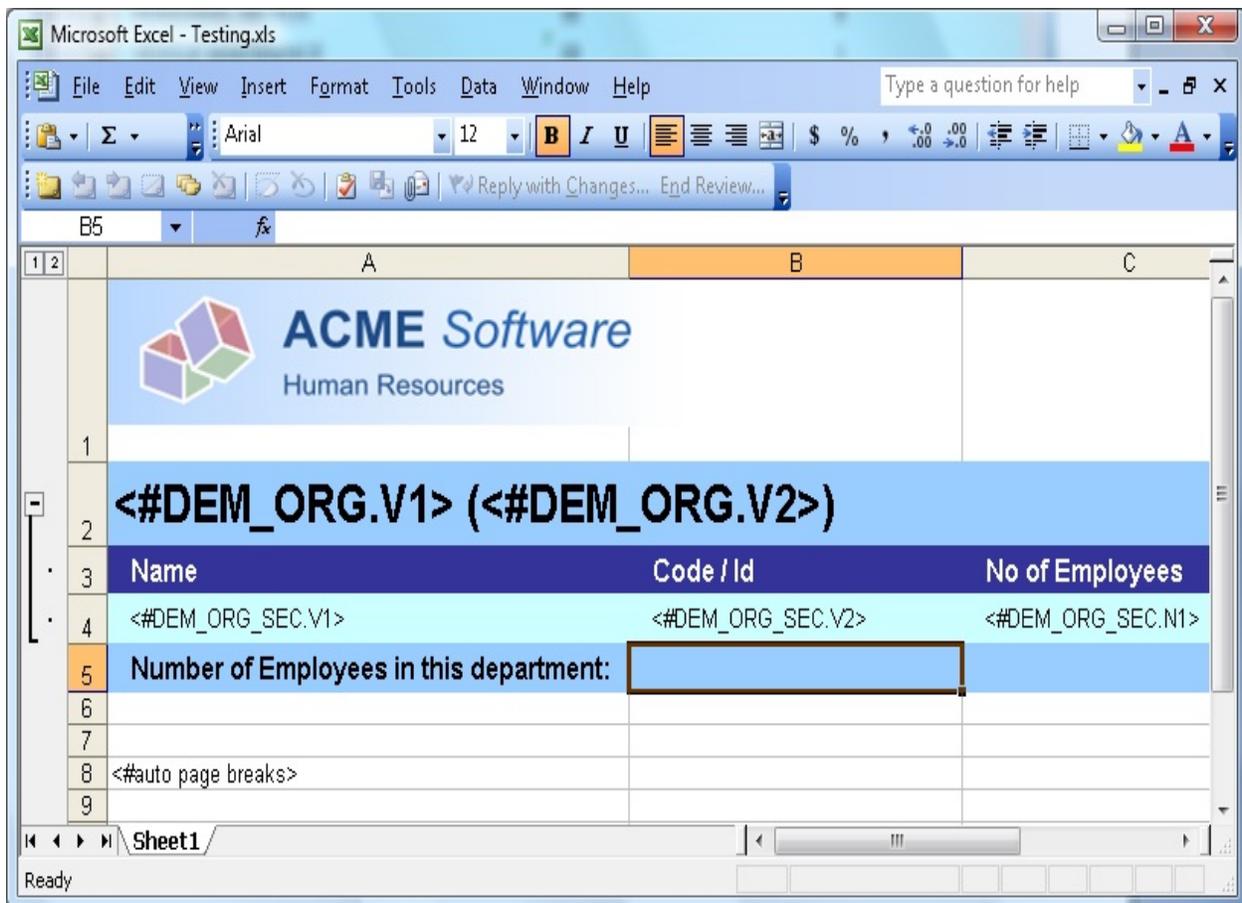
Below is an example of a report generated from your template:



The screenshot shows a Microsoft Excel spreadsheet with the following data:

ACME Software Human Resources		
ADMINISTRATOR DEPT (ADM)		
Name	Code / Id	No of Employee
INTERNAL ADMIN SRV	01	12
PURCHASING SECTION 2	02	18
ACCOUNTING SECTION	03	17
SALES & MARKETING	04	12
MAINTENANCE	05	10
PERSONNEL SECTION	06	8
VEHICLE MAINTENANCE	09	1
Number of Employees in this department:		78

Now let's go through the steps of inserting a running total figure into the report template.



You would use the SUM function to get the total of the figures that would be inserted into column C in the generated report. It would then be logical to put **SUM(C4:C4)** formula in cell B5. That however will not work because in the final report the range will always cover the first row only. In order to get the range expanded properly, you will need to add an extra empty row after your section band (row 4), and change your formula to **SUM(C4:C5)**.

Microsoft Excel - Testing.xls

File Edit View Insert Format Tools Data Window Help

Type a question for help

Arial 10 B I U

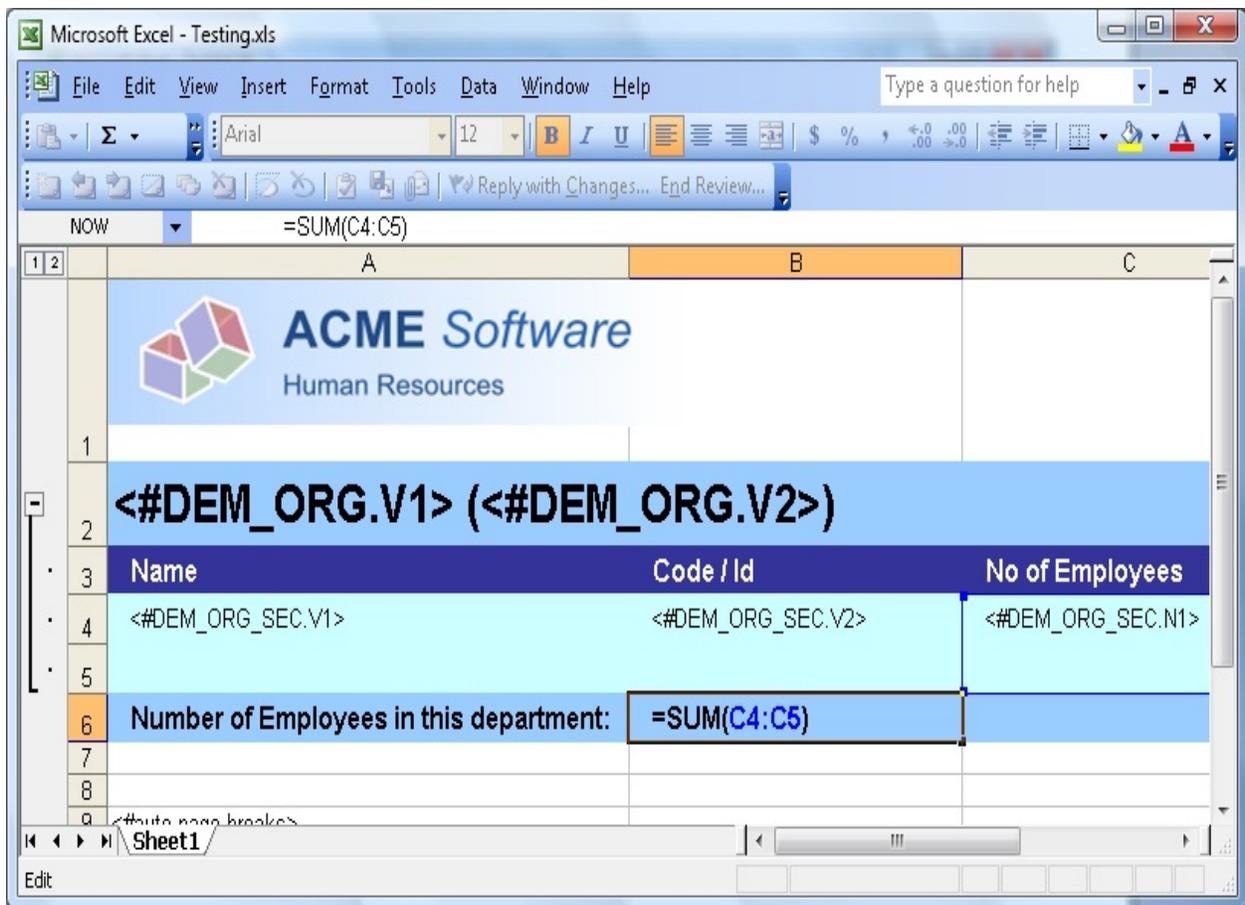
Reply with Changes... End Review...

A5

1			
2	<#DEM_ORG.V1> (<#DEM_ORG.V2>)		
3	Name	Code / Id	No of Employees
4	<#DEM_ORG_SEC.V1>	<#DEM_ORG_SEC.V2>	<#DEM_ORG_SEC.N1>
5			
6	<input type="checkbox"/> Number of Employees in this department:		
7			
8			
9	#auto org breaks		

Sheet1

Ready

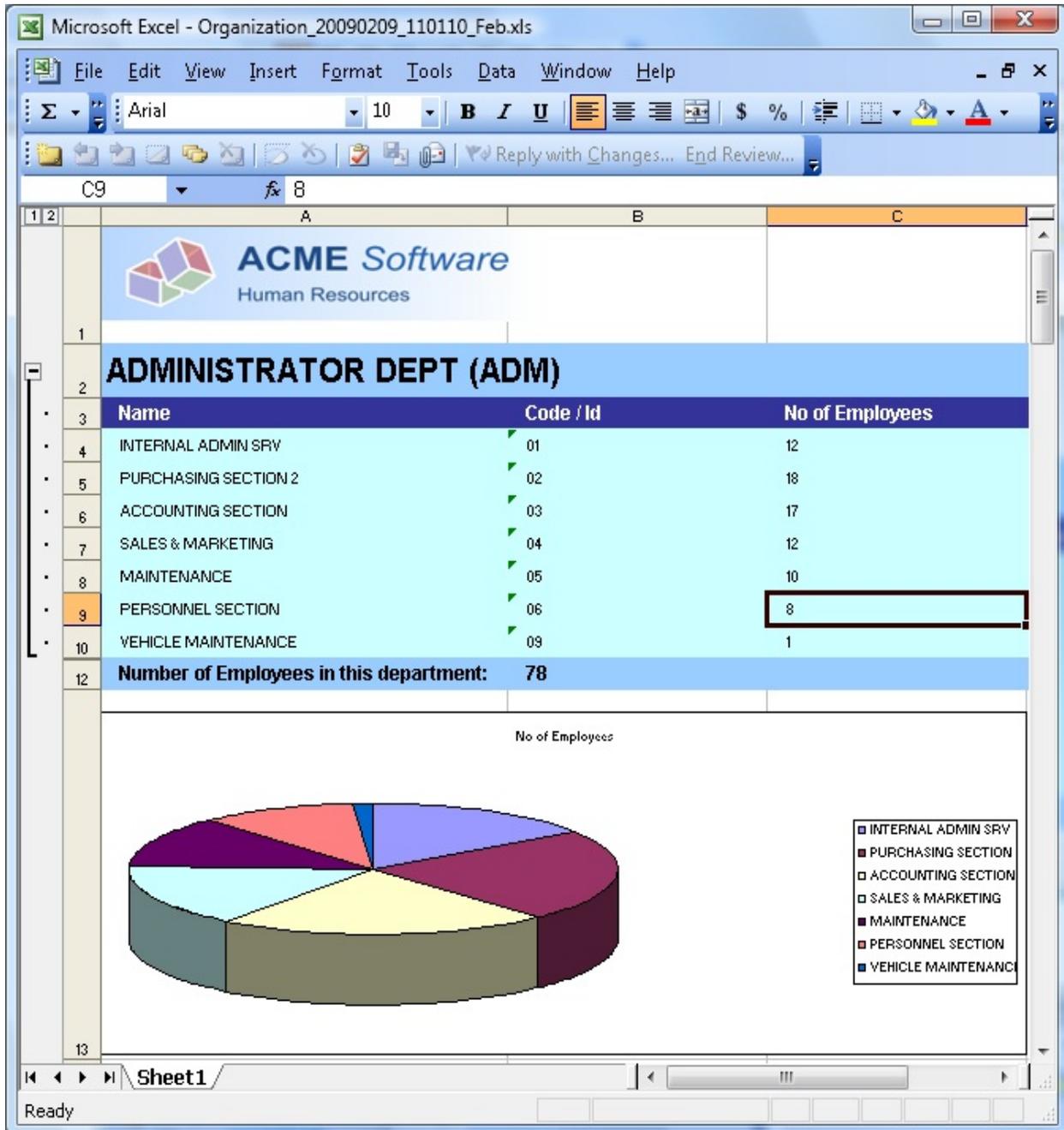


Hide the extra row to complete the process (right-click on the row header, select **Hide**).

Can I include charts in my report?

Yes you can. You just need to specify the chart range in the same way you do for the running total. Not every chart type is supported however if you are generating reports in PDF or HTML (generating Excel reports are not affected by this limitation).

Below is a sample of a report generated from a template with a chart.

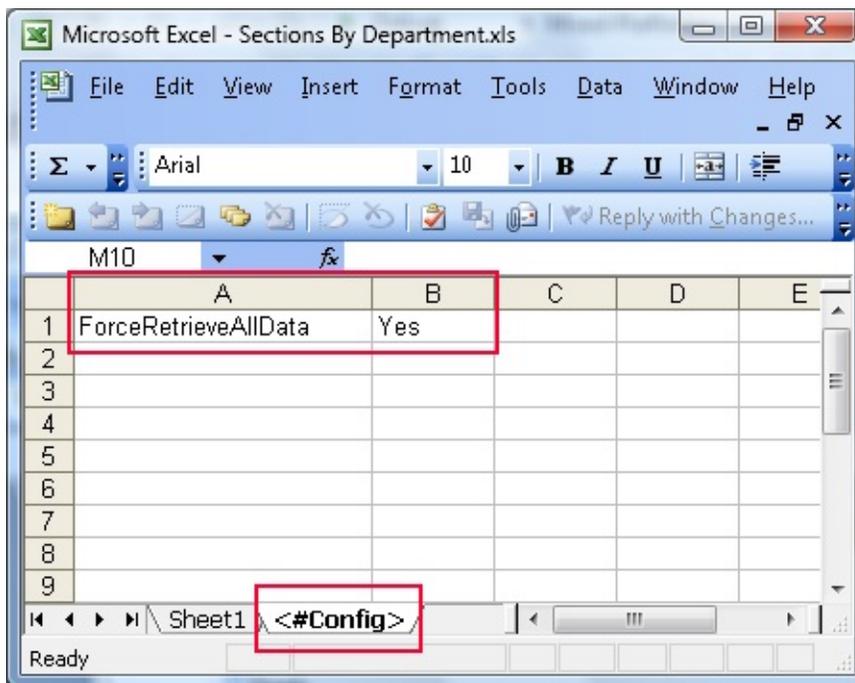


Is there a way to configure a template to force all data to be retrieved first prior to generating the report?

Yes.

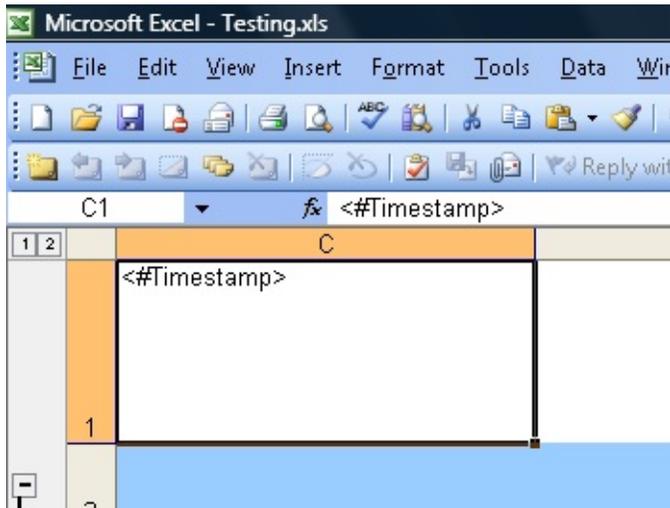
If you are using relationship handlers to retrieve data on-demand as the user expands nodes in the instance list, your report will only contain rows that have been retrieved from the server.

To configure a template to force all data to be retrieved first prior to generating the report, you need to create a 'configuration' sheet in your template. A configuration sheet is a special sheet called '<#Config>' that contains report settings. This sheet will not be included in the final report. Set the property **ForceRetrieveAllData** to **True** by putting the text **ForceRetrieveAllData** in column A1 and text **True** in column B1.

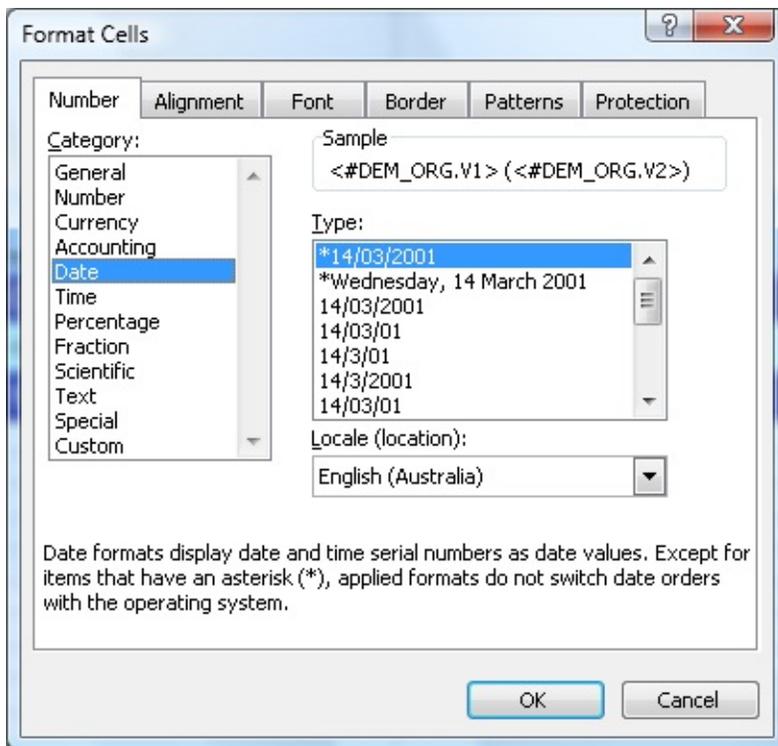


How can I put printing date & time on the report?

Insert the <#Timestamp> tag into your template.



To format the date or time the way you want, open the **Format Cells** dialog box (if you don't format the cell to use date or time format, the generated report will display it as a number).



How can I insert the username of the user who generates the report?

Insert the `<#User>` tag into your template.

How do I keep some rows together in a page?

Define a named cell range that covers the rows you want to keep together, and name the range **KeepRows_<Order>_<Name>**. You should substitute **<Order>** with a number and **<Name>** with any name (it's totally arbitrary, it's there just so that you can define multiple **KeepRows** range as Excel won't allow you to have two ranges with the same name). **Order** will almost always be 1 however there will be instances where you will want to put numbers greater than 1 for **Order**.

Example

Let's say that you have a template that produce the following report and you want to make sure that rows that make up a record are kept together in a page, so as to avoid the situation illustrated in the next figure.

The screenshot shows a PDF document titled "Employee_20090206_021740_Feb.pdf" in Adobe Reader. The document contains two employee records. The first record, for Paul Patrick (A1012), is highlighted with a red border. The second record, for Pattison, George (A1013), is also visible below it.

Employee Details	
PAUL, PATRICK (A1012)	
Address	6 Camillo Avenue SEVEN HILLS, NSW, 2147
Work Phone	222 2222
Home Phone	687 1717
Department	ADM
Section	01
PATTISON, GEORGE (A1013)	
Address	12 Augusta Avenue, PUNCHBOWL, NSW, 2016
Work Phone	212 3569
Home Phone	750 2562
Department	ADM
Section	01

Employee_20090206_022856_Feb.pdf - Adobe Reader

File Edit View Document Tools Window Help

1 / 12 54.9% Find

Employee Details

PAUL,PATRICK (A1012)

Address	6 Camillo Avenue SEVEN HILLS. NSW. 2147
Work Phone	222 2222
Home Phone	687 1717
Department	ADM
Section	01

PATTISON,GEORGE (A1013)

Address	12 Augusta Avenue, PUNCHBOWL. NSW. 2016
Work Phone	212 3569
Home Phone	750 2562
Department	ADM
Section	01

WOODS,BRADLEY (A1015)

Address	59 Darley Road, BEXLEY. NSW. 2030
---------	--

We want to avoid splitting this record

To make sure that the rows of a record are kept together, define a named range called **KeepRows_1_Employee** (the last part of the name Employee is arbitrary; you can call it anything you want, but you need to keep **KeepRows_1_** the same) as illustrated below.

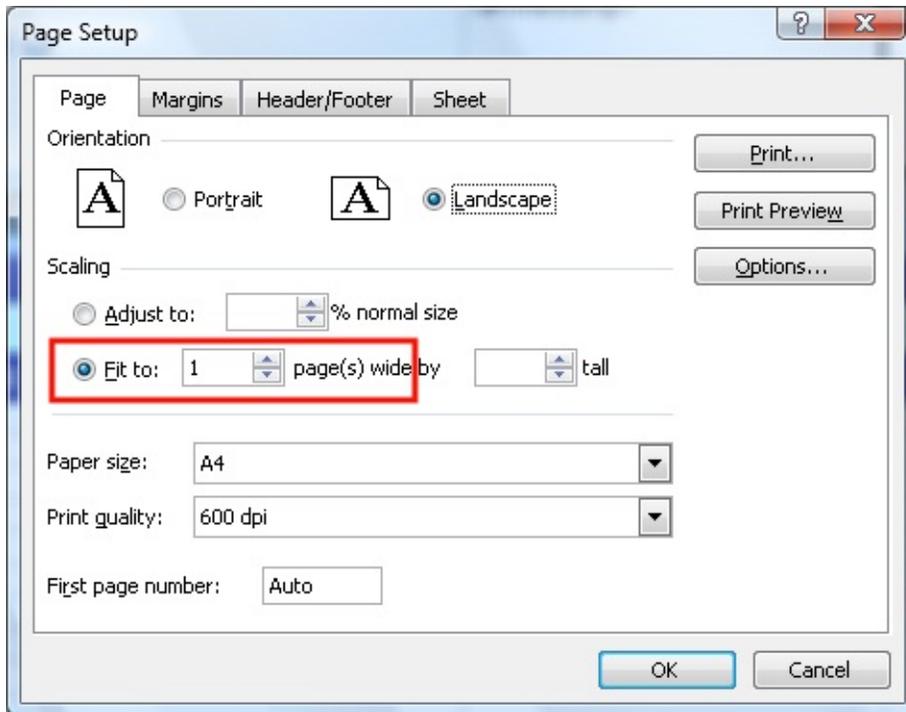
The screenshot shows the 'Define Name' dialog box in Microsoft Excel. The dialog box has a title bar 'Define Name' and a close button. It contains a list of names in the workbook: 'KeepRows_1_Emp', 'DEM_ORG_SEC_EMP', and 'KeepRows_1_Emp'. The 'Refers to' field contains the formula '=Sheet1!\$A\$2:\$A\$12'. The background spreadsheet shows a table with the following structure:

Employee Details	
	<#DEM_ORG_SEC_EMP.A1>
Address	<#DEM_ORG_SEC_EMP.A2>
	<#DEM_ORG_SEC_EMP.A3>
	<#DEM_ORG_SEC_EMP.N1>
Work Phone	<#DEM_ORG_SEC_EMP.A4>
Home Phone	<#DEM_ORG_SEC_EMP.A5>
Department	<#DEM_ORG_SEC_EMP.A6>
Section	<#DEM_ORG_SEC_EMP.A7>

Notice that the range only needs to cover the first column as columns are irrelevant here.

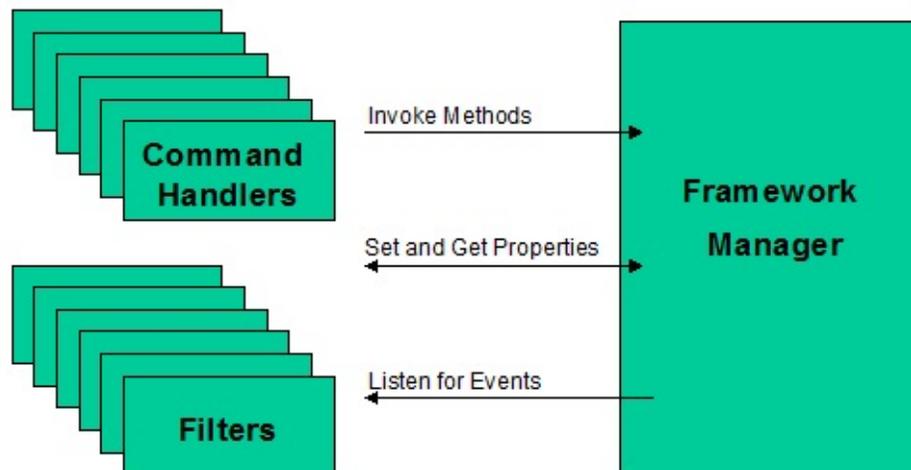
How can I tell a template to fit all columns in a page when generating reports?

Open the **Page Setup** dialog box, set to 'fit to 1 page width', then save your template.



Framework Manager

The Framework Manager (#AvFrameworkManager) is a component shipped with the Framework which provides a standardized set of services you can use in your filters and command handlers. You access Framework Manager services using methods, setting and getting properties and listening for events.



The services are:

- UpperCase Conversions
- Numeric to Alphanumeric Conversions
- Application Error Handling
- Framework Locking Service to Handle Unsaved Changes
- Saving unsaved changes using uQueryCanDeactivate / avNotifyDeactivation
- The Virtual Clipboard
- Basic Tracing Service
- Event Signaling Service
- Object Switching Service
- Custom Property Access Service
- User Authority Access Service
- Show Messages Service
- Temporarily Overriding Object Captions
- Get Visual LANSa Framework Icon Reference
- Change a visual style at run time

· [Framework Windows Management](#)

Also see [Programmatic server connection checking](#) .

UpperCase Conversions

Applies to Windows only.

The Framework manager provides an uppercase method.



For example:

```
Invoke #AvFrameworkManager.avUpperCase Value(#LastName)
```

will convert field #LastName to uppercase.

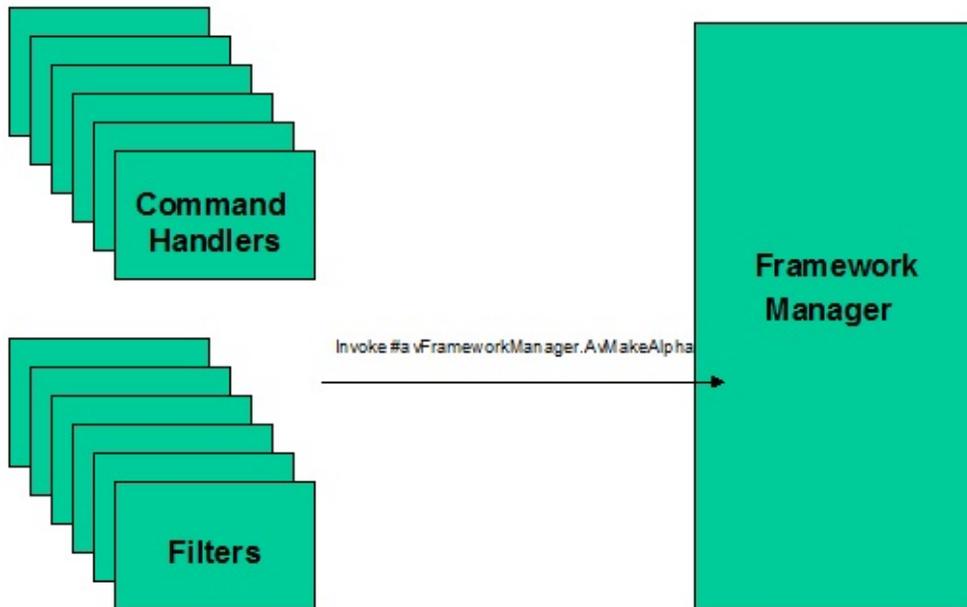
Note that with WAMs and other components enabled for Full RDMLX you can use the native Uppercase method instead:

```
#LastName := #LastName.uppercase
```

Numeric to Alphanumeric Conversions

Applies to Windows only.

The Framework manager provides a method for converting numeric values to alphanumeric form.



For example:

```
Invoke #AvFrameworkManager.avMakeAlpha fromNumeric(#ProdNo) Into(#ProdNoC)
Invoke #avListManager.AddtoList VisualID1(#ProdNoC) VisualID2(#ProdDes
```

will convert the numeric field #ProdNo to alphanumeric format in #ProdNoC (with 5 digits presented). #ProdNoC is then used as the alphanumeric visual identifier for an entry in the instance list.

Method avMakeAlphaValue supports output in various formats that are biased to the conversion of non-decimal identification data (eg: Product Numbers, Customer Numbers, etc) such as these examples:

FromNumericValue() Input Value	Output when Length() NOT specified	Output when Length(3) specified	Output when Length(7) specified	Output when Length(15) specified
-----------------------------------	--	--	--	--

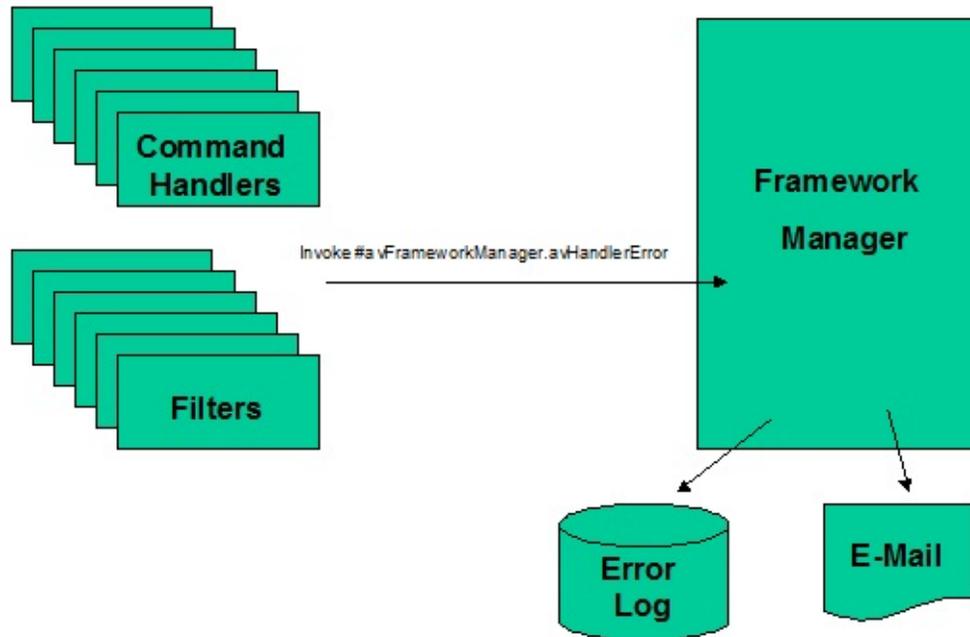
1	1	001	0000001	0000000000000001
345.678	345.678	345	0000345	0000000000000345
123456789	123456789	789	3456789	000000123456789
67-	67-	067-	0000067-	0000000000000067-

See [Visual Identifiers](#).

Application Error Handling

· Applies to Windows only.

The Framework services manager provides a standardized way for you to handle fatal errors in your application.



[How To Use It](#)

[Example](#)

[Properties Used](#)

How To Use It

You invoke the error handler in your application in this way:

```
Invoke Method(#AvFrameworkManager.avHandleError) CURR_COMP(*com  
CURR_RETC('<return code>')
```

Parameter description:

CURR_COMP(*component)	Leave unchanged, it will have the name of the component that had the error
CURR_CMD('<current command>')	Where current command is 20 byte long character field indicating the causing the error
CURR_ROUT('<routine name>')	Where routine name is 10 byte character long field indicating the routine name where the error has occurred
CURR_INFO('<Free formatted information about the error>')	Where Free formatted information about the error is a 255 byte long character field with any extra information you might think useful about the error
CURR_RETC('<return code>')	Where return code is a 2 byte long character field containing the return code, if applicable, of the command that caused the error.

Example

If your program had to locate the details of an employee and this information could not be found, you would have encountered a fatal error situation indicative of some sort of database corruption. In such a situation you might use the application error handler like this:

```
Fetch Fields(#SURNAME) From_File(PSLMST) With_Key(#EMPNO)
If_Status Is_Not(*OKAY)
  Invoke #AvFrameworkManager.avHandleError Curr_Comp(*component)
  Curr_Rout(ButtonHandler) Curr_Cmd(FETCH)
  Curr_Info('Employee details could not be accessed.')
  Curr_Retc(#io$Sts)
Endif
```

The standard error handler then does a number of things (depending up on how your Framework is configured) including:

- Displaying the fatal error details to the user.
- Recording the error details in a file.
- Emailing the error details to a nominated support email address.
- Terminating the Framework.

Properties Used

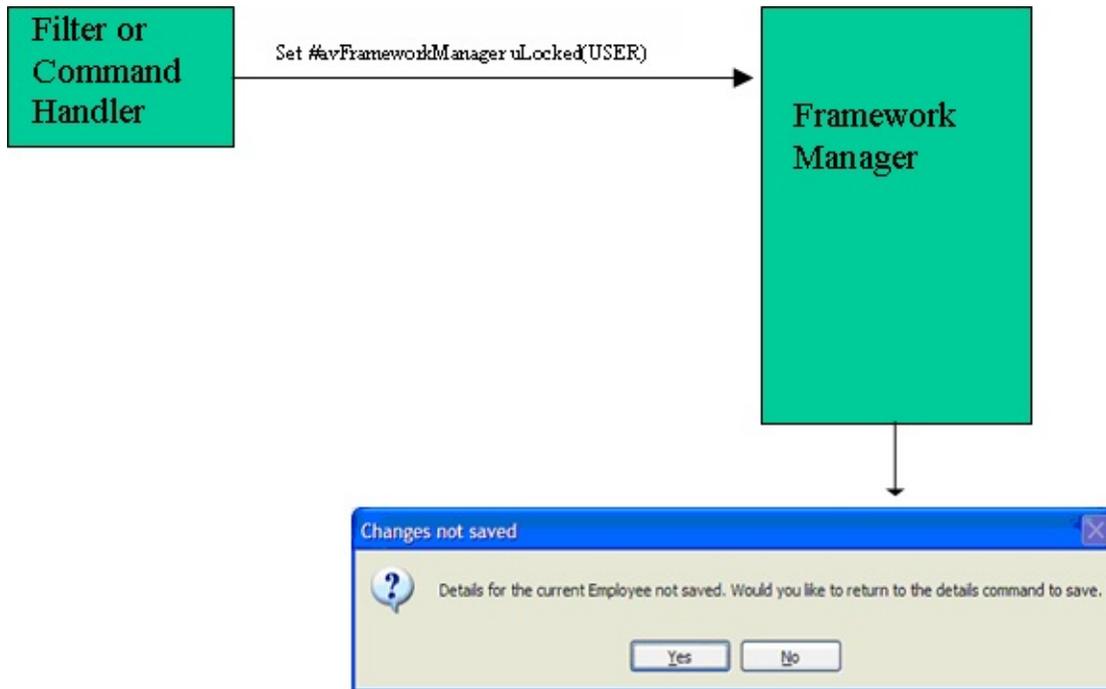
Address for Error Notification

Temporary Directory on PC

Framework Locking Service to Handle Unsaved Changes

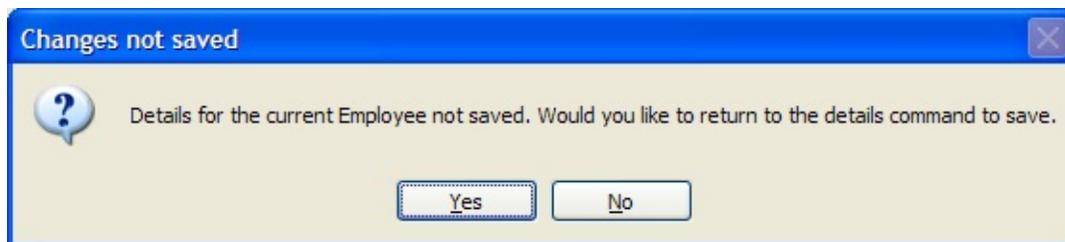
Applies to Windows.

The Framework manager provides a locking service that signals there are unsaved changes in a command handler or filter, and locks the Framework until the user has taken action to accept or cancel the changes.



A simple example of the use of a Framework lock is where a business object's details have been changed and the user then clicks on another instance of the same business object or invokes any other action in the Framework that would cause the changed details to be lost.

You can use the service to notify the user about unsaved changes and to lock the Framework when the business objects details change. A message is issued and the user has the choice of returning to the business object details to save them or continue with the requested action:



See the employee details command handler, DF_DET1, and the section details

command handler, DF_DET8, for complete examples of the use of the Framework locking service in Windows applications.

Framework locking is implemented through two Framework properties that are exposed to filters and command handlers, `uLocked` and `uLockedMessage`.

- `uLocked` Values are `FALSE`, `USER`, `PROGRAM` or `PROGRAM_EXIT`.
- `USER` means that the Framework is locked, but that the user can elect to end the lock.
- `PROGRAM` means that the Framework is locked and only a program can unlock it by setting this property to `FALSE`.
- `PROGRAM_EXIT` means that the Framework is locked except when exiting or closing down and only a program can unlock it by setting this property to `FALSE`.
- `uLockedMessage` This is the message to be shown to the user if they attempt to do something that would violate the lock state.

For example you could put this code in the `Changed` event of fields on a command handler to lock the Framework until the user has saved changes:

```
Set #avFrameworkManager uLocked(USER) uLockedMessage('Details for the current Employee not saved. Would you like to return to the details command to save.')
```

Note: Framework locks apply within the scope of each main Framework window. If the [Allow this Object to be Opened in a New Window](#) option is used to open a new main Framework window, the new window and all its child windows get their own (new) locking scope that is independent of all others.

Saving unsaved changes using uQueryCanDeactivate / avNotifyDeactivation

In version epc831 or later of the Framework it is possible to add a redefined method to command handlers called uQueryCanDeActivate.

This method is invoked when the end-users try to move from the command handler to somewhere else in the Framework, or when they try to close down the Framework. It is run when the user clicks on another command tab, a new instance of the business object or another business object or application.

It is particularly useful for saving unsaved changes. The routine can check whether changes need to be saved. If so, the user can be asked "Do you want to save your changes before continuing?" (Yes/No). If they answer Yes, their changes can be saved.

At the time the routine is run, the command handler still has all its values as they were before the user attempted to move away. So checking for unsaved changes, and saving those changes, is easy.

To use it, you need to set the avNotifyDeActivation property in the command handler's initialize routine

```
* Handle Initialization
```

```
Mthroutine Name(uInitialize) Options(*REDEFINE)
```

```
* Do any initialization defined in the ancestor
```

```
Invoke #Com_Ancesor.uInitialize
```

```
* Activate Check for unsaved changes (Unsaved changes logic)
```

```
set #Com_Owner avNotifyDeactivation(TRUE)
```

```
Endroutine
```

Then add a redefined uQueryCanDeActivate routine to your command handler:

```
* The Framework initiates this when the user moves to another command tab, or business object instance, or business object, or application, or closes the framework.
```

```

* (The framework may initiate this method multiple times)

MTHROUTINE NAME(uQueryCanDeactivate) OPTIONS(*REDEFINE)
* Define_Map For(*Result) Class(#vf_elBool) Name(#Allow)

#Allow := True

if '#pty_NeedsSaving *eq TRUE)'

* If something needs saving, ask the user if they want to save it

USE BUILTIN(MESSAGE_BOX_SHOW) WITH_ARGS('YESNO' 'YES'
*Default *Default 'The notes have been changed. Would you like to save your
changes before continuing?') TO_GET(#MSG_RET)

if '#MSG_RET *eq YES'

* Save everything
<< my save logic>>

endif

#pty_NeedsSaving := False

endif

endroutine

```

A more complicated version could set #Allow to false if there was an error during the save, and in that case the user would not go to where they clicked, (or the Framework would stay open if they were attempting to close it).

Also see:

[Reason Code](#)

[Comments/Warnings](#)

Reason Code

The optional ReasonCode parameter gives an indication of what the user is trying to do when the uQueryCanDeactivate check occurs. Note that one user action may result in multiple uQueryCanDeactivate checks.

Application Trace Shows		Code Description
VF_FPM09:MEEXECUTECOMMAND	1	Execute a command
VF_UM040:ITEMGOTFOCUSACCEPT	11	Main Instance List entry got focus
VF_UM083:ITEMGOTFOCUSACCEPT	12	Secondary Instance List entry got focus
VF_SY100:GET_LOCKSTATUS	13	Snap-in Instance List checking uCurrentLockStatus
VF_UM037.TABCHANGING	2	User clicked on a different tab
VF_UM014:ITEMGOTFOCUSACCEPT	21	Navigation pane (as two lists) application item selected
VF_UM015:APPVIEWBUTTONSBYID	22	Navigation pane (as two lists) – View button pressed
VF_UM015:ITEMGOTFOCUSACCEPT	23	Navigation pane (as two lists) business object selected
VF_UM016:UMENUITEMSELECTED	24	Navigation pane (as drop down) item selected
VF_UM016:ITEMGOTFOCUSACCEPT	25	Navigation pane (as tree view) item selected
VF_UM040.CLEAR_BUTTON	14	Clear the instance list
VF_UM003:CLOSEQUERY	3	Close a force-floated command handler
VF_UM046:CLOSEQUERY	4	Close a separate window command handler

VF_AC006:BEGINCLOSEFORM:A	31	Close Main Framework window
VF_AC006:BEGINCLOSEFORM:B	32	Close Secondary Framework window
VF_SY100:AVSWITCH	41	Switch
VF_SY150:USELECTAPPLICATION	26	Navigation pane application selected
VF_SY153.USETTHEME	43	User changes THEME
	999	Unknown

An action may result in multiple uQueryCanDeactivate checks, and multiple reason codes.

If you want to determine the reason code/s for a particular action, start an application level trace in the Framework and carry out the action. The trace will show you the action/s that occurred.

The codes have been grouped in this way:

- Command opening, Tab Opening, Floating Command Handler Closing: 1-9
- Instance List: 10-19
- Change of Application/Business Object: 20-29
- Framework Closing: 30-39
- Other Known: 40-49
- Unknown 999

Comments/Warnings

You must set `#Com_Owner.avNotifyDeactivation` to TRUE if you want to use it.

The `uQueryCanDeactivate` method may be invoked several times by a single user click, so it is important to reset `#pty_NeedsSaving` in the routine, so that subsequent invokes do not check with the user again.

There are warnings in the feature help about using method `uQueryCanDeactivate` and property `avNotifyDeactivation` in version epc831 of the framework. These warning can be ignored.

This method can be useful when dealing with objects that don't signal when they have changed. By using this method you can compare a snapshot of the object's current status with its as-loaded status, just at the point the user has finished with the command handler. (Rather than checking for changes every second).

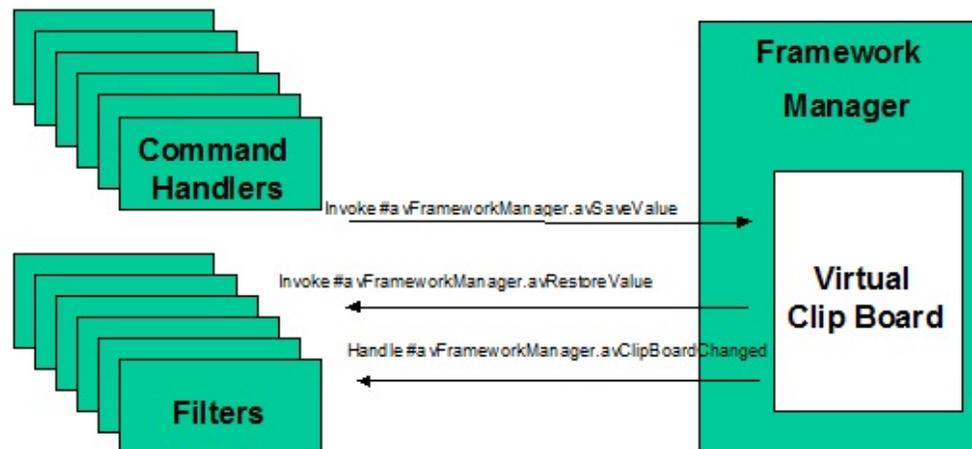
In VLF-WIN, if Framework locks seem to be released automatically after a number of minutes, check that the Framework's autosave interval is set to zero.

The Virtual Clipboard

Applies to Windows and **WAM**.

The Framework services manager provides a virtual clipboard facility. See the tutorial [VLF012WIN - Controlling Navigation Using Switching and the Virtual Clipboard](#) or [VLF012WAM - Controlling Navigation Using Switching and the Virtual Clipboard](#) for step-by-step instructions for how to use the virtual clipboard.

Note: Most of the facilities that the virtual clipboard provides are available in Windows or Web browser applications. The recommended way of learning to use the virtual clipboard is to conceptually understand this material from the Windows point of view.



The clipboard is designed to serve two main needs:

- For [Remembering Information](#) between application executions.
- For [Exchanging Information](#) between components.

To use the virtual clipboard most effectively you need to devise a standardized naming protocol for items that are posted onto it.

The clipboard uses a minimum 3 part naming standard and any name can support multiple instances (i.e.: become a list of values) and optionally function in multiple languages, so it's quite easy to devise an object.property style naming model for information that is posted to it.

For example, when a user starts using a General Ledger application they may

need to select the Company and a Currency that they will be working with. These values are accessed by lots of filters and command handlers within the application as the user moves around.

Conceptually, we now have a "GL Application" object that contains "SelectedCompany" and "SelectedCurrency" properties.

Using the clipboard you can easily define GLAPPLICATION.SelectedCompany and GLAPPLICATION.SelectedCurrency as things on the clipboard. Your GLAPPLICATION clipboard object is infinitely extensible without application recompiles and can contain lists, etc as required.

In both **Windows** and **WAM** browser applications you would code this naming standard as:

```
Invoke #AvFrameworkManager.avRestoreValue WithID1(GLAPPLICATION)
Invoke #AvFrameworkManager.avRestoreValue WithID1(GLAPPLICATION)
```

Equally you might devise a naming protocol for information that is private (i.e., scoped) within an individual function. For example, you might use PRIVATE.<function name>.<property name>. To save the value of #EMPNO using the naming standard as PRIVATE.<function name>.EmployeeNumber you might code this in a **Windows** or **WAM** browser application:

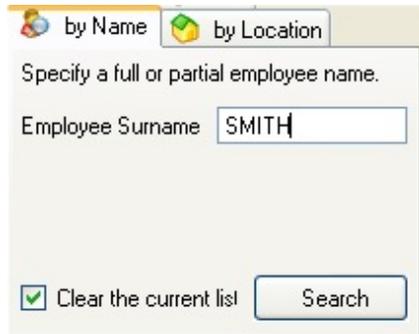
```
Invoke #AvFrameworkManager.avSaveValue WithID1(PRIVATE) WithID2(*C
```

See also:

- [Listening for Changes](#)
- [Other Things Worth Knowing](#)
- [Persistence, Resetting and Deploying in Windows Applications](#)

Remembering Information

Imagine that you had to create a filter like this and needed to remember what the last value entered in the surname field was:



You can do this by using the virtual clipboard.

If you execute this code whenever the surname is entered or changed:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(SURNAME) FromAVa
```

Then you have saved the current value of #Surname (say, "SMITH") onto the virtual clipboard using the symbolic name SURNAME.

To get the value back from the virtual clipboard you do this:

```
Invoke #AvFrameworkManager.avRestoreValue WithID1(SURNAME) ToAVa
```

When you put value "SMITH" onto the clipboard using the symbolic name SURNAME it is accessible to all the other command handlers and filters in your Framework.

Sometimes this is useful because you can pass information around between command handlers and filters.

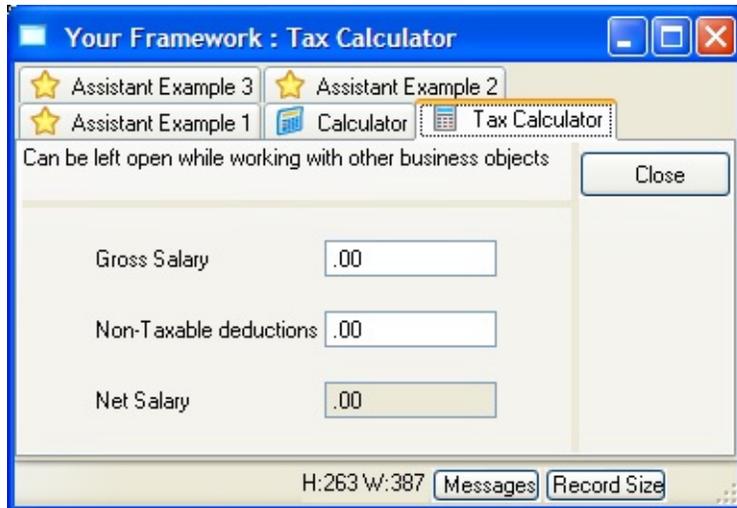
Sometimes this is not useful and you need to have private information on the clipboard. One way to do this is to use compound symbolic names for information that you put onto the clipboard. If we change the previous code samples to:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(*Component) WithID2  
FromAValue(#SurName)
```

Then the symbolic names used to identify "SMITH" on the virtual clipboard are now made up from the current component/function name and the identifier "SURNAME". In other words we have introduced a symbolic object-naming standard that will allow us to use the virtual clipboard in a more organized way. The best way to learn more about using the virtual clipboard to remember and share information, is to look at the Tutorials and the shipped Programming Techniques application.

Exchanging Information

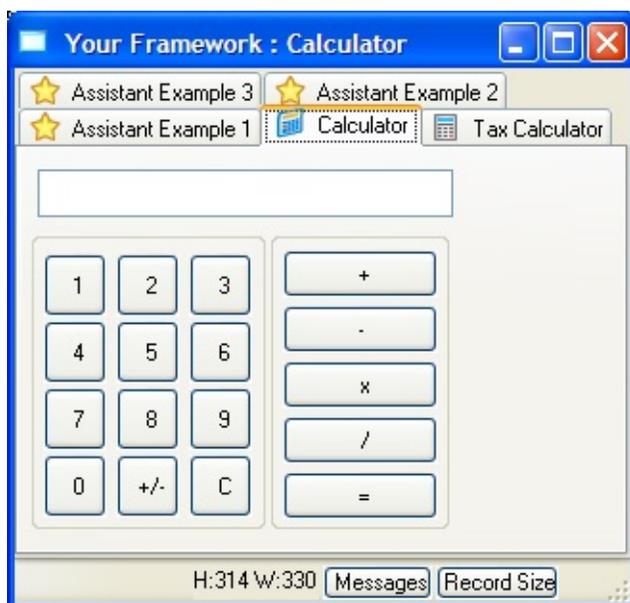
The shipped Tax Calculator (reusable part (DF_DET12)) uses the virtual clipboard to exchange information between components.



If you look at the source of the tax calculator you will see this command being executed at the end of the tax calculation routine to save the calculation result onto the virtual clipboard:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(Calculation) WithID2(  
    FromNValue(#DF_ELNET) Persistent(FALSE)  
    SignalChange(TRUE)
```

The normal calculator DF_DET13 also uses the clipboard:



After each calculation DF_DET13 executes this code to post calculation results:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(Calculation) WithID2(1)
    FromNValue(#VTOTAL) Persistent(FALSE)
    SignalChange(TRUE)
```

This is an example how to get net salary from the tax calculator:

```
EVTROUTINE HANDLING(#MITM_TAXCALC.Click)
    INVOKE method(#AvFrameworkManager.avRestoreValue) WITHID1(Calculation)
    SIGNAL event(SomethingChanged)
ENDROUTINE
```

This is an example how to get salary from the standard calculator:

```
EVTROUTINE HANDLING(#MITM_NORMCALC.Click)
    INVOKE method(#AvFrameworkManager.avRestoreValue) WITHID1(Calculation)
    SIGNAL event(SomethingChanged)
ENDROUTINE
```

Note the use of the UseNDefault(#Salary) parameter, to ensure that the current value is unchanged if no calculation can be found on the clipboard. What is being demonstrated by these examples is the exchange of information between the calculators and a form.

In summary both the tax and normal calculators post their latest calculation to the clipboard all the time.

This very simple concept is a foundation upon which more complex and elaborate information exchanges can be based.

Listening for Changes

Currently this part of the Virtual Clipboard feature is only available in Windows applications.

In the preceding section the use of the virtual clipboard for the exchange of information between command handlers was discussed. In the example the user had to use the right mouse button and select a pop-up menu option to cause the information exchange to take place (i.e.: it was manually initiated).

However, in some situations the virtual clipboard facility is most effective when the exchange of information happens without user intervention. When things are posted to the clipboard by the `avSaveValue` method you can use the special `SignalChange(TRUE)` parameter. The `TRUE` must be in uppercase. For example, the tax calculator (`DF_DET12`) posts its latest calculation like this:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(Calculation) WithID2(  
    FromNValue(#DF_ELNET) Persistent(FALSE)  
    SignalChange(TRUE))
```

In other words all listening filters and command handlers will be notified that something on the clipboard has been changed. It is up to all listening filters and command handlers to decide whether they are interested in the thing that has been changed.

In `DF_DET2` (New Employee) you will find a routine like this:

```
EvtRoutine Handling(#AvFrameworkManager.avClipBoardChanged) WithId1(  
    If '#Id1.Value = Calculation'  
        Change #Salary #NValue.value  
    Endif  
Endroutine
```

Here `DF_DET2` (New Employee) is listening to the system for clipboard changes.

When one occurs, the `Id1` value is immediately checked (both the calculators use a naming protocol that puts "Calculation" in the `ID1` field). If the clipboard value that has been changed is a calculation, then the new value is mapped into the salary field.

You can try this out by executing `DF_DET2` (New Employee) and `DF_DET12` (Tax Calculator) at the same time. As you do tax calculations they are

immediately and automatically mapped into the new employee's salary field. Again, this very simple example demonstrates a foundation upon which more complex and elaborate information exchanges can take place.

Other Things Worth Knowing

Note that:

- Clipboard information is normally remembered across executions of a Framework. You can stop this from happening by using `Persistent(FALSE)` when using the `avSaveValue` method.
- You should not put confidential or critical information on the clipboard.
- The volume and frequency of update of information on the clipboard will affect the performance of your application.

WithIdn

Information posted to the clipboard is assigned a set of identifiers.

In Windows applications these are the method parameters named `WithIdn`.

You can have up to 5 of these in Windows applications and 3 in Web browser application. Each one is a maximum of 32 characters long and they can contain concatenated data.

You need to define a protocol for using `WithIdn` values.

Generally `WithID1(*Component)` is a good way of making sure that a component's clipboard entries do not interfere with those of another component.

Of course, there are situations where you do want clipboard entries to be shared and sometimes your filters or command handlers are themselves used in multi-instance contexts. Refer to [The Virtual Clipboard](#) for details of sharing the clipboard entries.

ToAValue and FromAValue

Many of the examples use the `ToAValue` and `FromAValue` parameters for alphanumeric information. There are equivalent `ToNValue` and `FromNValue` parameters for working with numeric values.

The `ToAValue` / `FromAValue` information can be at most 255 characters long.

The `ToNValue` / `FromNValue` information is a 30,9 numeric value, but the maximum precision is normally constrained by the operating system to 15 digits (integer or decimal).

ReturnCode

Method `avRestoreValue` has a `ReturnCode` parameter that can be used to check whether a clipboard entry was found. For example:

```
Invoke #AvFrameworkManager.avRestoreValue WithID1(*Component) WithI  
If '#IO$Sts *ne OK'
```

```
Change #NumCopies 42
Endif
```

UseAValueDefault, UseAValueUDefault and UseNValueDefault

Method `avRestoreValue` has `UseAValueDefault`, `UseAValueUDefault` and `UseNValueDefault` parameters that can be used to specify which default value should be used when a clipboard entry cannot be found.

This means that the previous example could be more easily coded like this:

```
Invoke #AvFrameworkManager.avRestoreValue WithID1(*Component) WithI
```

ForLanguage

There is an additional clipboard parameter named `ForLanguage` that can be used when the thing being remembered is language dependent. Typically this is used like this:

```
Invoke #AvFrameworkManager.avSaveValue WithID1(*Component) WithID2
```

This means that the clipboard maintains different `#HelloMessage` values for different languages.

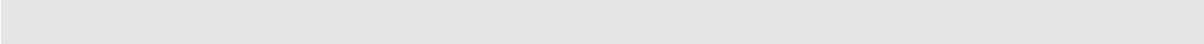
Instance

There is an additional numeric parameter named `Instance` which allows you to keep many instances of a value on the clipboard. This effectively allows you to place lists of things onto the clipboard.

For example, imagine you had a list view containing employee numbers (`#EmpNo`) and salaries (`#Salary`) in your program. Then this code:

```
Change #Instance 0
SelectList #ListView
  Change #Instance '#Instance + 1'
  Invoke #AvFrameworkManager.avSaveValue WithID1(*Component) WithII
    FromAValue(#Empno) Instance(#Instance)
  Invoke #AvFrameworkManager.avSaveValue WithID1(*Component) WithII
    FromNValue(#Salary) Instance(#Instance)
Endselect

Invoke #AvFrameworkManager.avSaveValue WithID1(*Component) WithID2
  FromNValue(#Instance)
```



Creates a list (and count) of employee numbers and salaries on the clipboard

Persistence, Resetting and Deploying in Windows Applications

In Windows-based Framework applications the virtual clipboard contents persist indefinitely.

How do the contents persist?

The virtual clipboard contents persist in files named PPP_User_Virtual_ClipBoard.Dat and PPP_Framework_Virtual_ClipBoard.Dat, where PPP is the current partition identifier. These files must reside in the users' temporary directory.

Can the clipboard file names or storage folder be changed?

Yes. If you create your own framework entry point(s) you can alter properties **uVCFolder** and **uVCFilePrefix** to specify the folder and/or file names used to store virtual clipboard data.

Refer to comments of the shipped version of UF_EXEC, UF_DESGN, UF_ADMIN and UF_DEVEL for more information.

How can the contents be reset or reinitialized?

You can reset the contents of a virtual clipboard by shutting down the Framework and deleting the files PPP_Virtual_ClipBoard.Dat and PPP_Framework_Virtual_ClipBoard.Dat from the temporary directory.

If you are logged on as developer you can do this by using the menu option (Framework) -> (Virtual Clipboard) -> Delete clipboard contents at exit. As you shut down the Framework the virtual clipboard files will be deleted instead of being saved.



When the Framework is restarted it will be unable to find the clipboard files so the virtual clipboard is reset to being empty.

However, if a file named VF_User_Virtual_Clipboard_Default.dat and/or VF_Framework_Virtual_Clipboard_Default.dat exist in the LANSA partition execute folder they will be used to re-initialize the clipboard to a shipped set of values.

Deploying Clipboard Initial Values

If you want to define a set of default values for the clipboard files that are

shipped to end-users for use the very first time they log on, or when they reset/reinitialize their clipboard you need to create and deploy files VF_User_Virtual_Clipboard_Default.dat and/or VF_Framework_Virtual_Clipboard_Default.dat.

To make these files use the developer menu options (Framework) -> (Virtual Clipboard) -> Save as default like this:

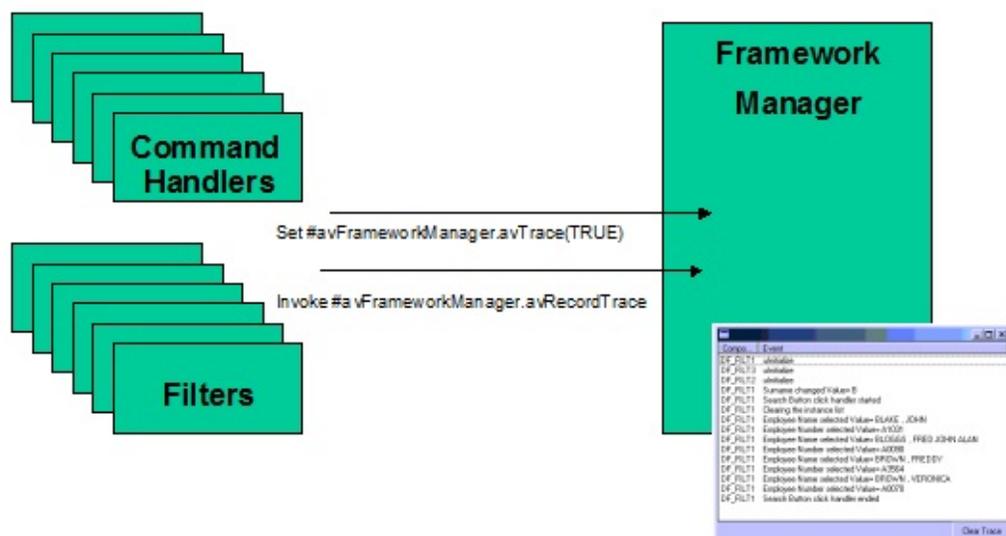


This causes your current clipboard to be saved as VF_User_Virtual_Clipboard_Default.dat and VF_Framework_Virtual_Clipboard_Default.dat in your partition execute directory, ready for inclusion into your deployment package.

Basic Tracing Service

Applies to **Windows** and **WAM**.

The Framework manager provides a basic tracing service to help you locate problems in your filters or command handlers.



The tracing service can be used in conjunction with, or independently of, the normal LANSa application debugging and tracing facilities.

See tutorial [VLF014WIN - Debugging/Tracing](#) or [VLF014WAM - Debugging/Tracing](#) for step-by-step instructions of how to trace Framework applications.

In Windows and WAM Applications

In **Windows** applications you can add tracing capabilities to your applications by using one or more of the three available `avRecordTrace` methods:

`avRecordTrace` for general tracing:

```
Invoke #AvFrameworkManager.AvRecordTrace Component(#Com_Owner)
Event('The search button was clicked')
```

`avRecordTraceAValue` for tracing alphanumeric values:

```
Invoke #AvFrameworkManager.AvRecordTraceAValue Component(#Com_Ov
AValue(#Empno) Event('Employee selected')
```

`avRecordTraceNValue` for tracing numeric values:

```
Invoke #AvFrameworkManager.avRecordTraceNValue Component(#Com_Ov
NValue(#Salary) Event('Salary found'))
```

The avRecordTrace methods will do nothing at all until tracing is turned on. This generally means that tracing operations can be left in your application with only minor performance overheads.

Tracing is turned on and off programmatically by setting the avTrace property:

```
Set #AvFrameworkManager avTrace(TRUE)
Set #AvFrameworkManager avTrace(FALSE)
```

If you are using the Framework as a designer you can turn tracing on or off by using the (Framework) -> (Tracing) -> Application Level or System Level menu options at any time.

To trace values when a WAM first starts use a #avFrameworkManager.uInitialize event handler. Similarly to trace values as a WAM closes use a #avFrameworkManager.uTerminate event handler:

Trace values when WAM starts:

```
EvtRoutine Handling(#avFrameworkManager.uInitialize)
Options(*noclearmessages *noclearerrors)

Invoke #AVFRAMEWORKMANAGER.avRecordTrace
Component(#com_owner) Event('WAM Filter is initialising')
Endroutine
```

Trace values when WAM ends:

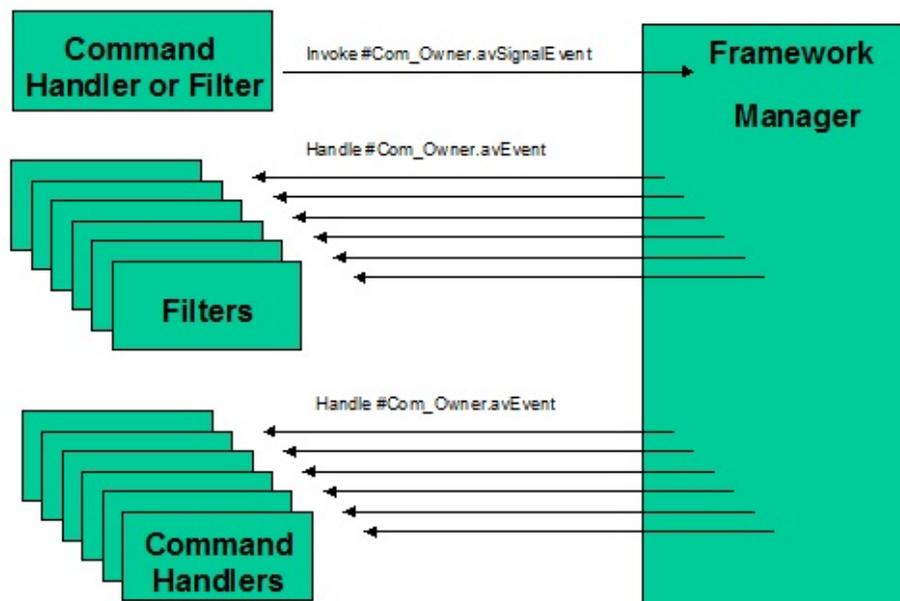
```
EvtRoutine Handling(#avFrameworkManager.uTerminate)
Options(*noclearmessages *noclearerrors)

Invoke #AVFRAMEWORKMANAGER.avRecordTrace
Component(#com_owner) Event('WAM Filter is terminating')
Endroutine
```

Event Signaling Service

Applies to **Windows** and **WAM**.

The Framework manager provides a simple to use event signaling service that may be used in **Windows** or Web browser applications:



Refer to tutorial [VLF013WIN - Signaling Events](#) or [VLF013WAM - Signaling Events](#) for step-by-step instructions of how to use signalling.

When something significant happens in your filter or command handler you may choose to signal an event to other active filters or command handlers so that they can take appropriate action.

To make event-processing work you need two things:

- A filter or command handler needs to signal the event. This is called **publishing** the event.
- Other filters or command handlers need to listen for the event. This is called **subscribing** to the event.

Additional information may be sent along with the event. This is called the **payload**.

In Windows Applications

To **publish** an event in a Windows application you use the method `avSignalEvent`. For example:

```
Invoke #Com_owner.avSignalEvent WithId(Employee_Deleted) SendAInfo1(?
```

This example signals an event identified as "Employee_Deleted". It includes the current value of field #EMPNO as additional payload string 1, presumably to identify the employee that has been deleted.

To **subscribe** to (or listen for) an event in a Windows application you need to have an EvtRoutine in your filter or command handler. To do this you listen for an avEvent signal like this example:

```
EvtRoutine Handling(#Com_owner.avEvent) WithId(#EventId) WithAInfo1(#  
  If '#EventId.Value = Employee_Deleted'  
  
  Change #EMPNO #AInfo1.Value  
  
  << Handle the event >>  
Endif  
Endroutine
```

This example specifically subscribes to the event identified as the "Employee_Deleted" event. The payload protocol used for this event says that the employee number will be received in payload additional information string 1.

In WAM Applications

To **publish** an event in a WAM application you use the method avSignalEvent. For example:

```
Invoke #ThisHandler.avSignalEvent WithId(UPDATE_LIST_ENTRY) SendA
```

To **subscribe** to (or listen for) an event in a WAM application you need to first register the event like this example;

```
Invoke Method(#avFrameworkManager.avRegisterEvent)  
Named(UPDATE_LIST_ENTRY) Signalaswamevent(2)
```

Then add an event handler as in the following example;

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_2)  
Withid(#eventid) WithAinfo1(#Ainfo1) Options(*noclearmessages  
*noclearerrors)
```

```

If '#EventId.Value = Update_List_Entry'

Change #EMPNO #AInfo1.Value
fetch #xg_ident pslmst with_key(#empno)

* Set up the visual Identifier(s)

#UF_VisID1 := #EMPNO
#UF_VisID2 := #SURNAME
Use BConcat (#UF_VisID2 #GIVENAME) (#UF_VisID2)

* Add instance details to the instance list

Invoke #avListManager.AddtoList Visualid1(#UF_VisID1)
Visualid2(#UF_VisID2) AKey1(#EMPNO)
Endif

Endroutine

```

General Considerations

- Having events flying around in your application impacts its performance. You need to assess this impact in an environment that reflects your worst-case deployment environment. Make sure that you do not use too many events or do too much processing when handling an event. Your event model needs to fit into the resource that you have available to execute it in. This applies doubly to Web browser applications where event handling causes client-server interactions to occur.
- The basic payload that can be sent along with an event allows for up to 5 alphanumeric strings and 5 numeric values. If this is not enough or if you want to send lists of information use the virtual clipboard. The shipped Programming Techniques application contains an example of sending lists of information along with an event by using the virtual clipboard.
- You should think about a simple and consistent naming standard for your events. Do not use event identifiers that contain periods (".") to avoid conflicts with events issued by the Framework. Avoid special characters that may cause code page issues with different execution platforms.

- Look at the shipped "Programming Techniques" application for examples of how to code and use events in Web browser applications.

For more detailed information about event handling see:

- [avSignalEvent](#)
- [#Com_owner.avEvent](#)

avSignalEvent

- WithId** An event identifier you assign. An event identifier can be up to 32 characters in length.
- SendAInfo1-
SendAInfo5** These allow up to 5 additional alphanumeric strings to be sent along with the event. The protocol used to assign these parameters to events is at your discretion. Maximum length 32 characters.
- SendNInfo1-
SendNInfo5** These allow up to 5 additional numeric values to be sent along with the event. The protocol used to assign these parameters to events is at your discretion. Maximum precision 30,9.
- Wait** Indicates whether the event should be processed by the listeners before control is returned by the avSignal Event method. Allowable values are TRUE or FALSE. The default value is FALSE. Use TRUE with caution !
- To** Indicates the scope in which the event should be signaled. The default value FRAMEWORK indicates that every active command handler and filter in the Framework should be notified of this event. The other allowable value is BUSINESSOBJECT, which indicates that event should only be signaled within the current business object. If you use BUSINESSOBJECT in a non-business object context, then it is treated as if you had specified FRAMEWORK.
- WindowScope** When Framework windows have been opened by the ‘Open in a New Window...’ menu option, this parameter may be used to control the overall scope of the event being signaled.
CURRENT, which is the default value, indicates the event should only be signaled to applications belonging to the current window.
MAIN indicates the event should be signaled in the scope of the main window only.
ALL indicates that the event should be signaled into all active windows.
The only other allowable value is the name of a specific window.

#Com_owner.avEvent

Some things worth knowing about handling the avEvent event are:

WithId	Identifies the event. Structurally a good way to use code like this: <pre>Eventroutine Handling(#Com_owner.avEvent Options(*NOCLEARMESSAGE Change #vf_eIdn #EventId.Value Case #vf_eIdn When '= Event1' << Execute subroutine or invoke When '= Event2' << Execute subroutine or invoke When '= Event3' << Execute subroutine or invoke ... etc etc etc ... When '= Eventn' << Execute subroutine or invoke EndCase Endroutine</pre>
WithAInfo1 -> WithAInfo5	Use these to receive the SendAInfo1 -> SendAInfo5 when the event was signaled.
WithNInfo1 -> WithNInfo5	Use these to receive the SendNInfo1 -> SendNInfo5 when the event was signaled.
Sender	Use this to determine the name of the component

	<p>event. This is most commonly used like the following:</p> <pre> EvtRoutine Handling(#Com_owner.avEvent) * If this event was not signaled by this If '#Sender.value *ne #Com_Owner.N: << handle the event >> Endif Endroutine </pre>
<p>#Com_Owner.avFilterActivated and #Com_Owner.avHandlerActivated</p>	<p>Filters and Command Handlers have properties that are currently considered to be activated or deactivated.</p> <ul style="list-style-type: none"> • #Com_Owner.avFilterActivated (in a filter) • #Com_owner.avHandlerActivated (in a command handler) <p>These properties contain strings TRUE or FALSE. They allow logic in a filter or command handler to be currently in an activated state. Typically the events that are signalled should be ignored.</p> <p>A filter or command handler is activated if its corresponding property may be able to interact with it.</p> <p>A filter or command handler in a minimized window may still be considered to be active.</p>

You should only check for events that your program is interested in. Other events should be ignored as quickly as possible.

As you are implementing your application you may find it useful to maintain a table of events like this example:

Event Id	Notified To	Waits for Completion	Additional Alphanumeric Information	Additional Numeric Information

	FRAMEWORK BUSINESSOBJECT	TRUE / FALSE	1= 2= 3= 4= 5=	1= 2= 3= 4= 5=
	FRAMEWORK BUSINESSOBJECT	TRUE / FALSE	1= 2= 3= 4= 5=	1= 2= 3= 4= 5=
	FRAMEWORK BUSINESSOBJECT	TRUE / FALSE	1= 2= 3= 4= 5=	1= 2= 3= 4= 5=

Object Switching Service

Applies to **Windows** and **WAM**.

See the tutorial [VLF012WIN - Controlling Navigation Using Switching and the Virtual Clipboard](#) or [VLF012WAM - Controlling Navigation Using Switching and the Virtual Clipboard](#) for step-by-step instructions for how to use the object switching service

This service allows your filters and command handlers to switch control between different business objects and to execute commands at the Framework, application or business object level.

The target business object must be able to be selected from the menu (the option Allow selection from the navigation pane in the target business object properties should be checked, and the user should be authorised to the business object), at the time the switch occurs. Switching mimics the actions that a user would perform.

The switching service is invoked by using method `#avFrameworkManager.avSwitch` with the appropriate parameters.

[Introductory avSwitch Examples](#)

[Referencing Applications, Business Objects and Commands](#)

[Advanced avSwitch Examples](#)

[avSwitch Method](#)

[AvAddSwitchInstances Event](#)

[AvAddSwitchInstance Method](#)

[Troubleshooting Switch Examples](#)

Introductory avSwitch Examples

Note that avSwitch can only call Framework applications. You cannot use it to call external applications such as Notepad.exe.

AvSwitch Method Examples for Windows or WAMs

Description

```
Invoke #avFrameworkManager.avSwitch  
To(Framework)  
Execute(Tax_Calculator)  
Caller(#Com_Owner)
```

Switches control to the Framework and executes the command handler associated with the command named "Tax_Calculator".

```
Invoke #avFrameworkManager.avSwitch  
To(Framework)  
Execute(*EXIT)  
Caller(#Com_Owner)
```

Switches control to the Framework and executes the command handler associated with the command named "*EXIT". This switch would cause the Framework to close down. It is handled exactly as if the user selected "File" then "Exit" from the menu bar.

```
Invoke #avFrameworkManager.avSwitch  
To(Application)  
Named(GeneralLedger)  
Caller(#Com_Owner)
```

Switches control over to the application named "GeneralLedger". No command is executed.

```
Invoke #avFrameworkManager.avSwitch  
To(BusinessObject)  
Named(Customers)  
Caller(#Com_Owner)
```

Switches control over to the business object named "Customers". No command is executed.

```
Invoke #avFrameworkManager.avSwitch
```

Switches control over to the

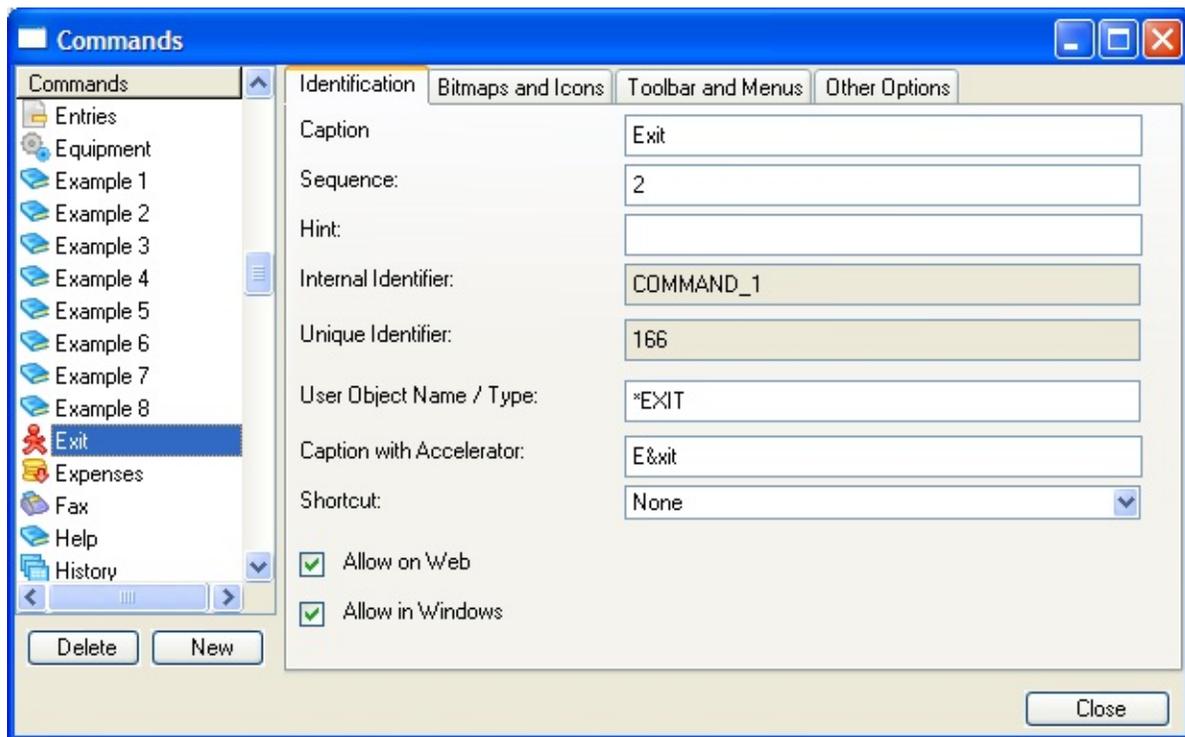
To(BusinessObject)
Named(Customers)
Execute(New)
Caller(#Com_Owner)

business object named
"Customers" and then executes
the command named "New".

Referencing Applications, Business Objects and Commands

In the preceding examples many names are used to identify applications, business objects and commands (eg: Tax_Calculator, GeneralLedger, New, '*EXIT'). These names are not the captions of the applications, business objects or commands, because captions change frequently and are multilingual.

The names are the User Object Name/Type values associated with the applications, business objects or commands. For example, if you look at the Identification tab of the definition of the Exit command:



Here the command's Caption is 'Exit' (at least in English) and it's User Object Name / Type is '*EXIT'. To reference the Exit command in a switch operation you should use the value '*EXIT' (the user object name/type), not 'Exit' (the caption).

Advanced avSwitch Examples

In the shipped Programming Techniques application Switching, command handler DF_DET17 displays the employees that work in the currently selected section:

The screenshot shows the Switching application window. The main window has a menu bar (File, Edit, View, Help, Windows (Framework) (Administration)) and a toolbar with icons for New, Reports, Web Site, About, Address, Resources Organization, Bookings, and Charges. A Quick Find search box is also present. The left sidebar shows a tree view with 'Programming Techniques' expanded to 'Switching'. The main area displays a table of employees:

Department	Section	Phone	Address	
ADM	01	INTERNAL A...	679 2536	2:
ADM	02	PURCHASIN...	952 6475	2:
ADM	03	ACCOUNTIN...	560 3633	2:
ADM	04	SALES & MA...	364-8905	2:
ADM	05	MAINTENANCE	(02) 456-7896	2:
AUD	01	ADMINISTRA...	411 8616	2:
AUD	02	PURCHASING	858 5002	2:
AUD	03	ACCOUNTING	728 6949	2:

Below the table is a 'Clear List' button. A modal dialog box titled 'Switching : Switch to Employees (ADM-02)' is open, showing 'Total Employees that work in this Section : 3' and a table of employee details:

Employee Number	Employee Surname	Employee Given Name(s)
A1002	SMYTHE	JOHN
A1005	SMITHS	PETER
A1014	MOORE	JOHN

The dialog box also has a 'Show Details' button. The status bar at the bottom shows 'Messages', 'Ready', 'Local', 'ENG', 'VLFPGMLIB', '21/08/11', and '17:09'.

You can select employee(s) from the displayed list and switch over to show their details by clicking the Show Details button. The display of the selected employee(s) details is achieved by using the avSwitch method. Please refer to the source code of DF_DET17 for a complete explanation of how the avSwitch method is used.

avSwitch Method

Method: avSwitch

Parameters:

Name	Usage	Class	Description
To	Input - Mandatory	Alpha – max length 32	Specify as one of FRAMEWORK, APPLICATION or BUSINESSOBJECT indicating the object to which control is to be switched.
Named	Input – Optional	Alpha – max length 32	Specifies the User Object Name/Type of the APPLICATION or BUSINESSOBJECT that control is to switch to.
Execute	Input – Optional	Alpha – max length 32	Specify the User Object Name/Type of any command that is to be executed in the target FRAMEWORK, APPLICATION or BUSINESSOBJECT.
ClearInstanceList	Input – Optional	Boolean	Clear the current business object instance list before adding new instances with the avAddSwictInstance method. Specify as TRUE or FALSE (in uppercase) only. Default value is FALSE. NOTE: This parameter is only effective if new instances are added with the avAddSwictInstance method.
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).

CallerInfo	Input – Optional	Alpha – max length 20	Use this optional argument to provide additional identification information.
TargetWindow	Input – Optional	Alpha – max length 256	Specifies the target window in which the switch operation should be performed. Allowable values are CURRENT (the current window), MAIN (the main window) or specific window name. The default value is MAIN.

Examples:

Refer to shipped example command handler DF_DET17.

AvAddSwitchInstances Event

Event: avAddSwitchInstances

Parameters:

Name	Usage	Class	Description
Caller	Received – Optional	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
CallerInfo	Received - Optional	Alpha – max length 20	Use this optional argument to provide additional identification information.

Examples:

Refer to shipped example command handler DF_DET17.

AvAddSwitchInstance Method

Method: avAddSwitchInstance

Parameters:

Name	Usage	Class	Description
VisualID1	Input – Optional	Alpha – max length 32	This is the first visual identifier of the business object instance. Alphanumeric.
VisualID2	Input - Optional	Alpha – max length 50	This is the optional second visual identifier of the business object instance.
AKey1 AKey2 AKey3 AKey4 AKey5	Input - Optional	Alpha – max length 32	These are the optional alphanumeric programmatic identifiers of this business object instance. The identification protocol used for the identifier is at your discretion.
NKey1 NKey2 NKey3 NKey4 NKey5	Input - Optional	Numeric – max (15,0) precision	These are the optional numeric programmatic identifier of this business object instance. The identification protocol used for the identifier is at your discretion.

Examples:

Refer to shipped example command handler DF_DET17.

Troubleshooting Switch Examples

The following table should be used as a checklist for resolving switch issues:

OK Check

The To() parameter is FRAMEWORK, APPLICATION or BUSINESSOBJECT.

The Named() parameter refers to the User Object Name/Type of the correct application or business object. It does NOT refer to the Caption.

The Execute() parameter refers to the User Object Name/Type of the correct command. It does NOT refer to the command's Caption.

The command identified by the Execute() parameter is enabled for the Framework, application or business object. Display the Commands Enabled tab of the Framework, application or business object involved. Verify that the command to be executed is shown in the Enabled list.

The parameter Caller(#Com_Owner) has been specified

In all avAddSwitchInstances event handlers an initial check like this has been made to ensure that the event should be handled by the current component:

```
If_ref #Caller is_not(*Equal_to #Com_Owner)
```

```
    Return
```

```
Endif
```

See shipped example DF_DET17 for examples.

In web browser applications executing with trace mode on will show the switch flow operations and may indicate the cause of the error.

Custom Property Access Service

The Framework services manager provides a facility to access [Custom Properties](#).

In Windows applications custom property values are retrieved by using the `avGetUserProperty` method.

Method `avGetUserProperty` Parameters

Name	Usage
AtLevel	Use this parameter to specify the level at which the custom property is defined. Allowable values are F (Framework), A (Application) or B (Business Object). The default value is F (Framework) level.
WithName	Use this parameter to specify the name of the custom property. The special property value <code><name>.Count</code> can be used to obtain a count of how many instances of the
Instance	Use this parameter when a property has multiple values (i.e. instances) to specify which instance should be returned. The default value is 1.
AlphaValue	Use this parameter to specify the field or component into which any alphanumeric value that the property has should be returned. The custom property must be defined as type alphanumeric for this parameter to be used effectively.
NumericValue	Use this parameter to specify the field or component into which any numeric value that the property has should be returned. The custom property must be defined as type numeric for this parameter to be used effectively.
BooleanValue	Use this parameter to specify the field or component into which any Boolean value that the property has should be returned. The custom property must be defined as type Boolean parameter to be used effectively.

Examples of retrieving Custom Property valuesExample 1

Retrieve the current value of Framework level alphanumeric property named PRINTER:

Windows

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Withname(PRINTER)
```

Example 2

Retrieve the 3rd instance of an application level numeric property named COMPANYNUMBER:

Windows

```
Invoke Method(#avFrameworkManager.avGetUserProperty) AtLevel(A)  
Withname(COMPANYNUMBER)  
Instance(3) NumericValue(#Company)
```

Example 3

Find out how many instances of a business object level property named STATE exist:

Windows

```
Invoke Method(#avFrameworkManager.avGetUserProperty) AtLevel(B)  
Withname(STATE.COUNT) NumericValue(#State_Tot)
```

User Authority Access Service

The Framework services manager provides a facility to access the current user's authority to a nominated object. For example, this service can be used to help a component decide whether to enable or disable buttons or switches. It is available only to Windows applications.

In Windows applications the current user's authority to an object is retrieved using the `avCheckUserAuthority` method.

Method `avCheckUserAuthority` Parameters

Name	Usage
To	Mandatory. Use this parameter to specify the type of the object to be checked. Allowable values are APPLICATION, BUSINESSOBJECT and COMMAND. Values may be specified in any case.
Named	Mandatory. The User Object Name / Type of the object to be checked.
InContext	Optional and required only for commands. Use this parameter to specify The User Object Name / Type of the object the command is associated with. Defaults to the current object.
ReturnValue	Mandatory. Returned as true or false. True means the user has authority to use the object and false means the user doesn't.

Examples of Authority Checks on Objects

Check the authority of the current user to the Demo application and return the answer to the variable `#STD_BOOL`:

```
Invoke Method(#avFrameworkManager.avCheckUserAuthority) To(APPLICA
```

Check the authority of the current user to the Organization business object and return the answer to the variable `#STD_BOOL`:

```
Invoke Method(#avFrameworkManager.avCheckUserAuthority) To(BUSINES
```

Check the authority of the current user to the 'Details' command of the Resources business object and return the answer to the variable #STD_BOOL:

```
Invoke Method(#avFrameworkManager.avCheckUserAuthority) To(COMMAN)
```

Check the authority of the current user to the 'Details' command of the current object and return the answer to the variable #STD_BOOL.

```
Invoke Method(#avFrameworkManager.avCheckUserAuthority) To(COMMAN)
```

Show Messages Service

The Framework services manager provides a facility to programmatically show the current set of messages.

In Windows applications the message box showing the current set of messages is displayed by invoking the `#Com_Owner.avShowMessages` method. It acts exactly as if the user had pressed the Messages button on the status bar.

Example of Showing Messages

Show the current set of messages:

Windows

```
Invoke Method(#Com_Owner.avShowMessages)
```

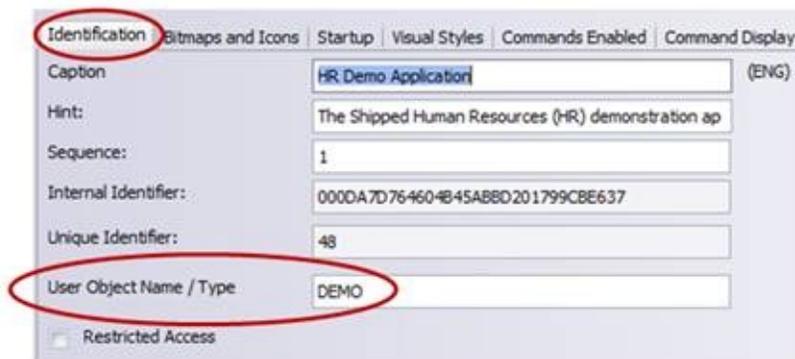
Temporarily Overriding Object Captions

Applies to: WINDOWS only.

The Framework Manager's `avOverrideCaptions` method allows you to temporarily override the captions associated with Applications, Views, Business Objects or Commands.

You can use this feature to show additional information about the state of an object in its caption, for example by indicating a count of how many associated messages exist or how many open orders were found.

Objects are identified by their unique system or user assigned User Object Name / Type found on their Identification tab:



For example this code changes the basic caption of the DEMO application:

```
Invoke Method(#avFrameworkmanager.avOverrideCaptions)  
Ofuserobjecttype(DEMO) Caption('HR Demo Application (5 messages)')  
Fim(#Fim)
```

Which causes the shipped DEMO application's caption to change from this:



To this:



Some things worth knowing:

- Caption overrides are not persistent. When a VLF window is closed the overrides are lost.
- Caption overrides apply everywhere (and only) within the scope of the window that issues the change request.
- Changing the alphabetical position of a caption may or may not change its relative position.

The parameters of the `avOverrideCaptions` method are:

<code>OfUserObjectType</code>	The unique system or user assigned type that identifies the object.
<code>Caption</code>	The new caption to be applied to the object.
<code>CaptionSingular</code>	The new singular caption to be applied to the object. This caption may not apply to all object types.
<code>CaptionAccel</code>	The new caption with an accelerator to be applied to the object. This caption may not apply to all object types.
<code>FIM</code>	The framework instance manager that identifies your window. Always pass as <code>FIM(#FIM)</code> .

Get Visual LANSAs Framework Icon Reference

The Framework services manager allows command handlers or filters to retrieve the reference to a VLF icon. It is available only to Windows applications.

If using a reference to an icon multiple times, it is better to use the `avFindIcon` method once in the initialization routine, and store the reference in a dynamic variable defined at the component level. (See example)

The reference to a VF_ icon is retrieved using the `avFindIcon` method.

Method `avFindIcon` Parameters

Name	Usage
Named	Mandatory. The name of the icon to be retrieved.
Reference	Mandatory. Returned as a reference to an object of class <code>PRIM_ICON</code> .

Example of using the `avFindIcon` method

Set a list item's icon to the Visual LANSAs Framework icon named `VF_IC386`

```
* Component definitions
DEFINE_COM CLASS(#PRIM_ICON) NAME(#ICON_NORM)
REFERENCE(*DYNAMIC)
...
* Initialization routine
invoke #avSystem.avFindIcon Named(VF_IC486)
Reference(#ICON_NORM)
...
* Use the reference
set #ltvw_1.CurrentItem Image(#ICON_NORM)
```

Change a visual style at run time

If you need a visual style to change at run time, you can swap in a new style or styles from any command handler, filter, or snap in instance list using this logic:

```
#avFrameworkManager.avSubstituteVisualStyle Ustyle(#MYSTYLE_A)
Uasname('VF_VS106')
#avFrameworkManager.avSubstituteVisualStyle Ustyle(#MYSTYLE_B)
Uasname('VF_VS101') Usignalchanged(True)
```

Where `uStyle(#MYSTYLE_A)` is a visual style that you have defined in the repository, and `'VF_VS106'` is one of the styles currently being used by the Framework.

If `uSignalChanged` is true the Framework will apply the changes and the replacement styles will be visible.

Method `avSubstituteVisualStyle` Parameters

Name	Usage
<code>uStyle</code>	Mandatory. The new style to be used
<code>uCaption</code>	Optional. The caption for the style (Default is the style name)
<code>uSuffix</code>	Optional. The suffix to be attached to the caption (Default is blank)
<code>uAsName</code>	Optional. The name of the visual style to be replaced (Default is the name of the new style)
<code>uSignalChanged</code>	Optional. Signal the VLF to apply the changes. (Default is false)

Framework Windows Management

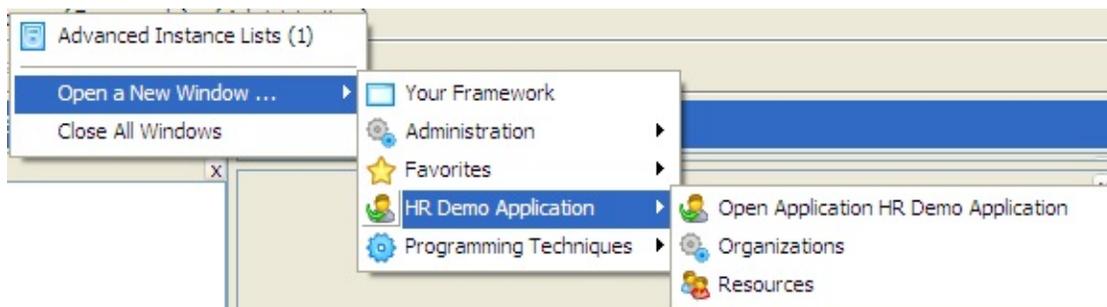
Programs and optionally end-users can open and control many Framework windows.

A Framework window contains a new instance of the Framework, or some subset of it.

For example, when you open a new Framework window it might contain:

- Another instance of the whole Framework
- Just a specific application (for example the Demo Application)
- Just a specific view of an application
- Just a specific business object (for example just Organizations or Resources)

If the Framework application design allows it, end-users can open new windows using the Windows menu and then Open in a New Window ... Framework menu options.



Or they can bring up the windows pop-up menu by right clicking anywhere.

The display of the Windows menus is controlled using the [Show the 'Windows' Menu in this Framework](#) option.

When multiple windows are open, a window control bar will appear under the tool bar of all the Framework windows:



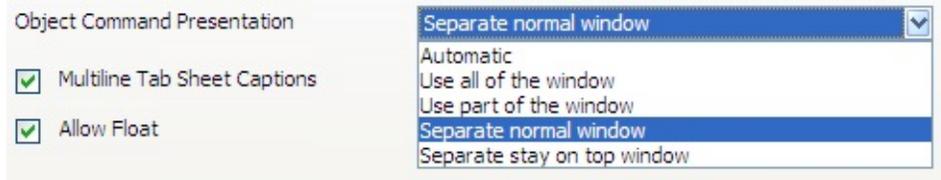
As a designer you can set a limit on how many windows that an end-user can have concurrently open using the [Number of Additional Windows a User can have Open Concurrently](#) option. You may control whether the whole Framework, individual applications, application views or business objects may,

or may not, be opened in independent windows using the object's [Allow this Object to be Opened in a New Window](#) property.

Allowing end-users to control their own desktops and work practices regarding independent windows make for very open and flexible applications. Trying too hard to programmatically control and manage what end-users do with independent windows may prove to be a complex, time consuming and ultimately fruitless task.

Notes

- Framework windows are available in Framework Windows applications only. This may constrain application design when complete application navigation portability needs to be identical in both the Windows and web environments. However, generally web users can achieve equivalent levels of functionality by opening multiple web browser instances or by using tabs within IE7.
- Please do not confuse Framework windows with the Object Command Presentation option in the Command Display tab which can be used to specify that a command handler is shown as a separate window.



[Programmatically Creating and Managing Windows](#)

[Notifying other Windows of Significant Events](#)

[Keeping Windows Open](#)

[Switching in Windows](#)

[Windows and Imbedded Interface Points](#)

[Windows Resource Usage](#)

Programmatically Creating and Managing Windows

Your programs can create and manage windows by calling the method `avShowWindow` in the Framework manager. For example:

Open a whole new instance of the Framework in another window named `MYWINDOW`:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)  
For(FRAMEWORK) WindowName(MYWINDOW)
```

Open the `DEMONSTRATION` application:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)  
For(APPLICATION) ofType(DEMO) WindowName(DEMO_WINDOW)
```

Open an application view named `HR`:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner) For(VIEW)  
ofType(HR) WindowName(DEMO_VIEW)
```

Open business objects `Organizations` and `Resources` in two independent windows:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)  
For(BUSINESSOBJECT) ofType(DEM_ORG) WindowName(DEMO_EMP)  
  
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)  
For(BUSINESSOBJECT) ofType(DEM_ORG_SEC_EMP)  
WindowName(DEMO_SECTION)
```

Remember that the `OfType(XXXXXXX)` names you specify on calls to `avShowWindow` are Object User Object / Type values specified on the Identification tab of the properties of the respective object:

User Object Name / Type

Using the avShowWindow method

When the avShowWindow method is invoked, it tests whether a window with the name specified exists.

If the named window already exists, it is activated (ie: restored from being minimized, if required, and brought to the front of all the Framework windows).

If the named window does not exist, it is created.

Then in both cases:

- The window's open information properties and object reference are updated with anything you supplied (see [Window Opening Information](#)).
- A switch operation is performed inside the window to any application or business you may have nominated.

In simple terms, you are saying to the Framework "Display a window with this name and pass this information into it, then cause it to switch to this application or this business object".

When you create a new window or switch an existing window it happens asynchronously to your program. So if you use avShowWindow and then immediately enumerate all open windows you will not find the window you just created (yet).

Window Names

You may have noticed from the preceding examples that windows have symbolic names. Here are some things you should know about window names:

- The names ALL, MAIN and CURRENT are reserved.
- When an end-user opens a window it is automatically assigned a unique name that starts with USER_. Do not rely on USER_ window names being the same from Framework signon to signon or from Framework version to version.
- Names are not end-user visible. They are programmatic names, case insensitive and may be up to 256 characters long. Being case insensitive means

they are often uppercased, so using just 'A' -> 'Z' and '0' -> '9' is advisable.

Window names are uniquely scoped and only addressable within a Framework process (ie: a LANSAX_RUN.EXE process). This means that if you start multiple X_RUN.EXE processes they can each contain a unique window named TESTWINDOW. An operation that involves signaling or switching window TESTWINDOW only refers to it within the current X_RUN.EXE process. No intra-process windows operations are currently provided.

[Finding a Window](#)

[Enumerating All Windows](#)

[The Current and Main Windows](#)

[Window Opening Information](#)

Finding a Window

You can find a specific window in your programs like this:

```
* Define a temporary class #VF_SY154 reference.
* Use VL's F2=Feature help to explore the properties
* and methods that class #VF_SY154 exposes for you to use.

Define_Com Class(#VF_SY154) Name(#Window) Reference(*Dynamic)

* Ask the Framework manager to locate a window by name and return a
reference

#Window <= #AvFrameworkManager.avWindow<'USER_EMPLOYEES'>

* Remember to handle the fact the window might be found (or not)

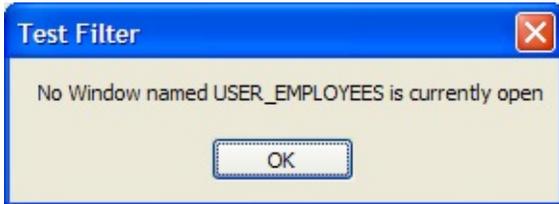
If (#Window *isNot *Null)
Use Message_box_Add ('The window named' #Window.WindowName 'is at
position' #Window.Top #Window.Left)
Else
Use Message_box_Add ('No Window named USER_EMPLOYEES is
currently open')
Endif

Use Message_Box_show

* Just to be absolutely sure, make the Window reference null to free it.
* You should never hang onto VF_SY154 references in your code.

#Window <= *Null
```

This code would display message boxes like these:



Important Note: Always free #VF_SY154 object references in your programs.

Enumerating All Windows

You can enumerate windows in your programs likes this:

- * Loop through all currently opened windows by using the
- * Windows collection of class VF_SY154 objects that the
- * Framework manager exposes for use in your programs
- * Use the VL F2=Feature Help to explore the properties
- * and methods that class #VF_SY154 exposes for you to use.

```
For #Window in(#AvFrameworkManager.avWindowCollection)
```

```
Use Message_box_Add ('The window named' #Window.WindowName 'is at  
position' #Window.Top #Window.Left)
```

```
EndFor
```

```
Use Message_Box_show
```

This code would display a message box like this:



Important Note: Always free #VF_SY154 object references in your programs.

The Current and Main Windows

In command handlers or filters, to get access to the current window or to the main window use properties `#Com_Owner.avCurrentWindow` or `#Com_Owner.avMainWindow`. These are immediately accessible `#VF_SY154` object references to the current and main windows respectively.

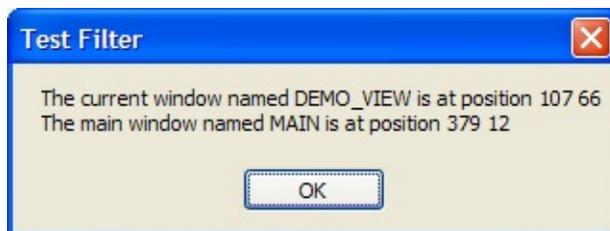
For example, if you put this code into any filter or command handler it will display the current and main window location:

```
Use Message_box_Add ('The current window named'  
#Com_Owner.avCurrentWindow.WindowName 'is at position'  
#Com_Owner.avCurrentWindow.Top #Com_Owner.avCurrentWindow.Left)
```

```
Use Message_box_Add ('The main window named'  
#Com_Owner.avMainWindow.WindowName 'is at position'  
#Com_Owner.avMainWindow.Top #Com_Owner.avMainWindow.Left)
```

```
Use message_box_show
```

Like this example:



Important Note: Always free `#VF_SY154` object references in your programs.

Window Opening Information

When a window is opened or activated by the `avShowWindow` method, information can be passed to it in the parameters `OpenInfo1` -> `OpenInfo5` and `OpenObjectRef`.

These parameters optionally allow 5 strings of up to 256 characters and an object reference to be used exchange information between windows.

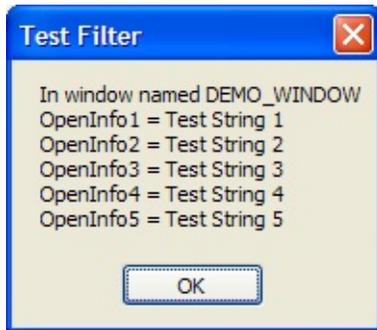
For example, if a filter or command handler opened a window named `DEMO_WINDOW` and passed 5 strings to it like this:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)
For(BUSINESSOBJECT) ofType(EMPLOYEES)
WindowName(DEMO_WINDOW) OpenInfo1('Test String 1')
OpenInfo2('Test String 2') OpenInfo3('Test String 3') OpenInfo4('Test String
4') OpenInfo5('Test String 5')
```

Then if a filter or command handler executing inside `DEMO_WINDOW` executed this code:

```
Use Message_box_Add ('In window
named'#Com_Owner.avCurrentWindow.WindowName)
Use Message_box_Add ('OpenInfo1
='#Com_Owner.avCurrentWindow.OpenInfo<1>)
Use Message_box_Add ('OpenInfo2
='#Com_Owner.avCurrentWindow.OpenInfo<2>)
Use Message_box_Add ('OpenInfo3
='#Com_Owner.avCurrentWindow.OpenInfo<3>)
Use Message_box_Add ('OpenInfo4
='#Com_Owner.avCurrentWindow.OpenInfo<4>)
Use Message_box_Add ('OpenInfo5
='#Com_Owner.avCurrentWindow.OpenInfo<5>)
Use message_box_show
```

The result would look like this:



You can get access to the opening information associated with any window by getting a class #VF_SY154 reference to the window (see preceding sections [Finding a Window](#) and [Enumerating All Windows](#)).

You can update the opening information currently associated with any window by setting the OpenInfo<> properties like this example, which updates the MAIN window:

```
#Com_Owner.avMainWindow.OpenInfo<1> := "Return String 1"  
#Com_Owner.avMainWindow.OpenInfo<2> := "Return String 2"  
#Com_Owner.avMainWindow.OpenInfo<3> := "Return String 3"  
#Com_Owner.avMainWindow.OpenInfo<4> := "Return String 4"  
#Com_Owner.avMainWindow.OpenInfo<5> := "Return String 5"
```

The OpenObjectRef property may be used to pass any type of object reference between windows. By doing this you can pass and share any amount or type of information between windows.

To get a filter or command handler in the new window to use the OpenInfo data, add a uShowWindowCompleted method routine to the command handler/filter. See [Notifying other Windows of Significant Events](#).

Notifying other Windows of Significant Events

The `avSignalEvent` method may be used to notify other windows of an event.

To control the scope of the signal use the `WindowScope()` parameter, which may be specified as `CURRENT`, `ALL`, `MAIN` or a specific window name.

The default value for this parameter is `CURRENT`.

Using the ‘Wake Up’ Method `uShowWindowCompleted`

The signaling approach works well when you know the window in question is open and listening for the event.

However, what if you don’t know whether the window in question is actually open? This situation is easiest to handle this way.

Imagine a command handler or filter in the `MAIN` window that executes this method:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)
For(BUSINESSOBJECT) ofType(EMPLOYEES)
    WindowName(EMP_WINDOW) OpenInfo1(#EMPNO)
```

What this says is "Show a window named `EMP_WINDOW` (creating it if you need to). It should contain just the `EMPLOYEES` business object. Set its opening information string to the value of `#EMPNO`".

Now imagine a filter (or command handler) that is either started, or already active, inside the window `EMP_WINDOW`.

It has this special ‘Wake up’ method in it:

```
Mthroutine uShowWindowCompleted Options(*Redefine)
```

```
#Empno := #Com_Owner.avCurrentWindow.OpenInfo<1>
```

```
Use message_box_show (ok ok info *Component #Empno)
```

```
Endroutine
```

This method is invoked every time some other window executes:

```
#AvFrameworkManager.avShowWindow  
WindowName(EMP_WINDOW)
```

In other words, this is a method that is saying ‘wake up, another window wants you to do something’. Typically the command handler or filter that is ‘woken up’ would use information passed in the open information strings to determine what it should do next.

This feature may be used to interlink high level business objects. For example imagine a window working with details named CUSTOMER_WINDOW and another working with order information named ORDER_WINDOW.

Scattered through the command handlers belonging to CUSTOMER_WINDOW you would find method calls like this:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)  
For(BUSINESSOBJECT)  
    ofType(ORDERS) WindowName(ORDER_WINDOW)  
    OpenInfo1(#OrderNumber)
```

Scattered through the command handlers belonging to ORDER_WINDOW you would find method calls like this:

```
#AvFrameworkManager.avShowWindow Caller(#Com_Owner)
```

```
For(BUSINESSOBJECT)
    ofType(CUSTOMERS)
WindowName(CUSTOMER_WINDOW)
    OpenInfo1(#CustomerNumber)
```

Orders and Customers are now interlinked making it very easy for an end-user to swap between the two applications.

Keeping Windows Open

Creating a new window is a relatively expensive operation. Often end-users close windows just make them 'get out of the way' and five seconds later they will often open the same window again, usually with different data content.

A simple solution to this problem is to use the parameter `KeepOpen(TRUE)` in the `avShowWindow` method or to change the `KeepOpen` property in class `#VF_SY154` window objects.

With `KeepOpen(TRUE)`, when end-users close a window, it is simply minimized just to get it out of their way. This means that in five seconds time when they want to display it again, it does not have to be created from scratch.

When you set a window to `KeepOpen(TRUE)`, it can only be closed when the end-user:

- Uses the signoff or exit command
- Closes the main window
- Uses the option 'Close All Windows - Except Main'

Or when a program changes it to `KeepOpen(FALSE)`.

You should never set the `MAIN` window to `KeepOpen(TRUE)`.

Switching in Windows

The use of multiple windows reduces the need to use classic switching.

However, you can request that a window switch to some other task by using the [avSwitch Method](#) . You can control which window is switched by using the `TargetWindow()` parameter, which may be specified as `MAIN`, `CURRENT` or a specific window name. The default value for this parameter is `MAIN`.

There are some rules about window switching that you need to be aware of:

- When a window is opened it has a scope.
- A window's scope includes all the things that can ever be used inside it.
- By limiting a windows scope the time taken to create it can be substantially reduced.

For example if you open a `FRAMEWORK` window then every application, business object and every command defined in the Framework is within its scope.

However, if you open a `BUSINESSOBJECT` window, then only the business object specified and any related application or Framework level commands are in the window's scope.

This means you cannot open a window named `TEST`, containing just business object A (say), and then later request that it switch to business object B. This is because business object B is not in window `TEST`'s scope.

You cannot legitimately switch a window out of its scope. You can try to do this and may even get it to do some 'interesting' things, but this type of switching is not supported by the Framework. Using it may produce unpredictable results, either now or in future Framework versions.

This also explains why the default value `TargetWindow(MAIN)` is used on the `avSwitch` method instead of `TargetWindow(CURRENT)`. The `MAIN` window has within its scope everything in the Framework, so the switch operation will

always be valid, at least from the scope perspective.

Once you start to modify `TargetWindow()` parameters values you need to think about whether the switch will be within the target windows scope.

Windows and Imbedded Interface Points

The system IIPs (Imbedded Interface Points) provide four methods that you can overload to perform special logic whenever the main window or a secondary window is opened or closed. These are called:

- avMAINWindowReady – Main window has opened and is ready for work
- avSECONDWindowReady – Secondary window has opened and is ready for work
- avCloseMAINWindow – Main window is to be closed
- avCloseSECONDWindow – Secondary window is to be closed

Refer to [Imbedded Interface Points \(IIPs\)](#) and the shipped version of reusable part UF_SYSTEM for more information about these methods and about how to implement you own IIPs.

Expanding, Shrinking and Focusing Panes

In the Windows Framework a filter or command handler can programmatically shrink the filter pane, the command handler pane, the instance list, or the navigate pane, and can also re-expand them. Command handlers can also programmatically expand themselves to occupy the space used by the filters and instance list (Maximize), and restore themselves back to their original size.

`avPaneShrink` and `avPaneExpand` methods are opposites. A shrunk pane can be expanded back to its original size using `avPaneExpand`. If a pane is not shrunk, then `avPaneExpand` will do nothing.

`avPaneMaximize` and `avPaneRestore` are opposites. A command handler can be expanded to use all the space used by the filter and instance list by using `avPaneMaximize`. When maximized, a command handler can be returned to its original size by using `avPaneRestore`.

`avPaneFocus` causes a pane to become expanded and then to receive the focus.

These methods are asynchronous – the request is queued up and waits for the command handler or filter that issued the instruction to finish its processing before the expand/shrink/maximize/restore is performed.

[avPaneShrink Method](#)

[avPaneExpand Method](#)

[avPaneMaximize Method](#)

[avPaneRestore Method](#)

[avCmdPanelState Property](#)

avPaneShrink Method

Method: avPaneShrink

Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
Pane	Input - Mandatory	Alpha – max length 32	Specify as one of FILTER, INSTANCE_LIST, NAVIGATE_PANE, COMMAND_HANDLER indicating the pane to be shrunk.

Examples:

```
invoke #avFrameworkManager.avPaneShrink Caller(#Com_Owner)  
Pane(FILTER)
```

avPaneExpand Method

Method: avPaneExpand

Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
Pane	Input - Mandatory	Alpha – max length 32	Specify as one of FILTER, INSTANCE_LIST, NAVIGATE_PANE, COMMAND_HANDLER indicating the pane to be expanded back to its original size after being shrunk.

Examples:

```
invoke #avFrameworkManager.avPaneExpand Caller(#Com_Owner)  
Pane(FILTER)
```

avPaneMaximize Method

Method: avPaneMaximize

Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
Pane	Input - Optional	Alpha – max length 32	Specify as COMMAND_HANDLER indicating the pane to be Maximized.

Examples:

```
invoke #avFrameworkManager.avPaneMaximize Caller(#Com_Owner)  
Pane(COMMAND_HANDLER)
```

avPaneRestore Method

Method: avPaneRestore

Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
Pane	Input - Optional	Alpha – max length 32	Specify as COMMAND_HANDLER indicating the pane to be restored back to its original size after a Maximize.

Examples:

```
invoke #avFrameworkManager.avPaneRestore Caller(#Com_Owner)  
Pane(COMMAND_HANDLER)
```

avPaneFocus Method

Method: avPaneFocus

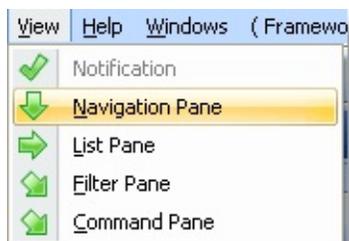
Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as Caller(#Com_Owner).
Pane	Input - Mandatory	Alpha – max length 32	Specify as one of FILTER, INSTANCE_LIST, NAVIGATE_PANE, COMMAND_HANDLER indicating the pane to be focused.

Examples:

```
invoke #avFrameworkManager.avPaneFocus Caller(#Com_Owner)
Pane(FILTER)
```

Refer to the View menu option in the shipped example Framework:



And to the shipped example hidden command handler DF_DET45 which implements these menu options.

Filters (class #VF_AC007), Command Handlers (class #VF_AC010) and Custom Instance List Browsers (class #VF_AC012) components may override ancestor method `uAcceptFocus` to specifically control which visible control receives the focus when `avPaneFocus` is used.

See shipped example components DF_FILT1 (filter), DF_T0023 (command handler) and DF_INST1 (custom instance list browser) for examples of this.

avCmdPanelState Property

Method: avCmdPanelState

Parameters:

Name	Usage	Class	Description
Caller	Input – Mandatory	PRIM_OBJT	Always specify this argument as avCmdPanelState<#Com_Owner>.

Examples:

```
#CPState := #avFrameworkManager.avCmdPanelState<#Com_Owner>
```

Windows Resource Usage

Usage Expectations

Resource Leakage

Usage Expectations

When using multiple Framework windows you need to consider resource utilization.

Every VLF window you open consumes additional resources. The amount of resource consumed depends significantly on the design of your application and the programming techniques you use. You need to balance resource utilization with resource availability on deployed systems.

There are some usage expectations you need to have about using Framework windows:

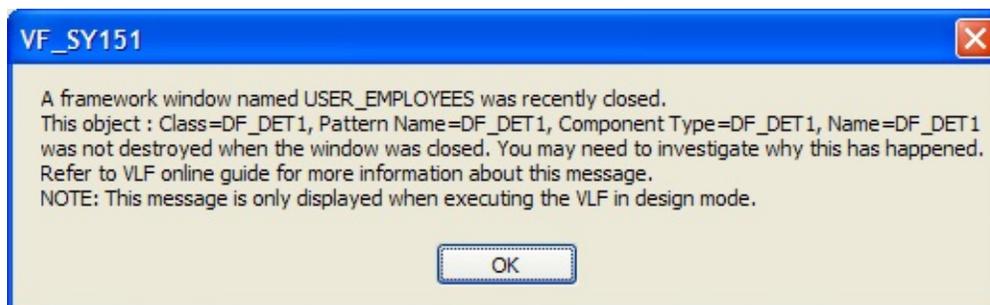
- It would be reasonable for a user to have two or 10 or even 20 windows open, but not 50 windows.
- Windows are heavyweight objects. They are intended to allow end-users to concurrently perform and switch between high level business tasks such as working with Customers, Products and Orders concurrently. They are not intended to handle simple lightweight tasks such as handling an F4 prompt.

Resource Leakage

A common resource utilization issue encountered in Windows applications is that when you close a window not all of the resources associated with it are freed up for reuse. This is called 'resource leakage' and if left unchecked it may eventually cause your application to fail or act unpredictably. To help you in this area the VLF includes a simple diagnostic tool to track some resource leakage at a high level.

These facilities are only activated when you use the VLF as a designer.

If you open and close a secondary Framework window and see a message like the example below, it indicates a resource leakage detected by the VLF:



This message is saying that a VL component (in this case a command handler instance named DF_DET1) was not destroyed when the window was closed.

In other words, DF_DET1 has 'leaked' resources. You should investigate the cause of this and try to prevent it. If you can't work out why this message is being displayed you should consult your Visual LANSA mentor for VL application design advice.

In VLF applications of simple to medium complexity you are unlikely to see this message. However as the sophistication of your VLF and VL application components increases you may encounter this message.

A common cause of this message is what is called a 'circular reference'. This is where one VL component (say, a command handler 'A') contains a reference to some other object (say, a reusable part named 'B') which itself contains a reference back to command handler 'A'.

The circular reference path may be even more complex. Component 'A' might refer to 'B' which refers to 'C' which refers to 'D' which refers back to 'A'. This is a much longer path, but it is still circular in nature.

Some rough guidelines for avoiding circular references:

- Try to avoid using Reference(*Dynamic) variables or object reference collections that are scoped across an entire BEGIN_COM/END_COM block (that is, across your whole VL program).
- When they are used, make sure that structured logic is in place (and used) to remove or nullify any object references that they may contain.
- When Scope(*Application) variables or collections are used to track object references make sure that logic is in place (and used) to remove the references again at the appropriate time.

In other words, imagine you are an object instance called 'A' and that other objects store references to you in their variables or collections. If you want 'A' to be destroyed, then you need to make sure all referencing objects drop all their references to 'A' first. Object 'A' will not be destroyed as long as any object holds a single reference to 'A'.

Important Disclaimer: This VLF facility is a diagnostic feature. It only tracks resource leakage in objects that are 'known' to the VLF runtime environment. As you create more sophisticated VLF applications they could in turn create and destroy many objects that the VLF runtime does not know about and therefore cannot track or report on. Just because you do not see the preceding message box it does not mean that your application is not leaking resources.

Advanced Filter Styles

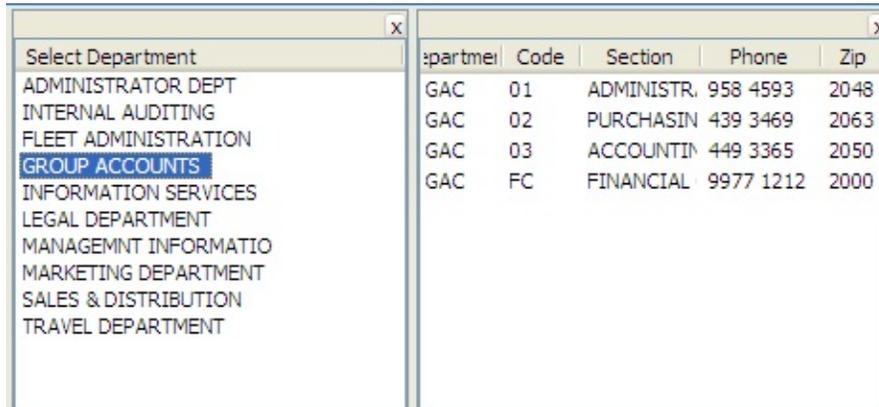
Filters are a powerful mechanism for building business object instance lists in many different ways.

The following section presents some ideas for advanced filter techniques that you might like to use in your application:

- [Instant Filters](#)
- [Drill Down Filters](#)
- [Power Filters](#)
- [Hidden Filter](#)
- [Mini Filters](#)

Instant Filters

An instant filter can, for example, present a list of all available departments. When you click on a department the sections business object instance list is instantly cleared and refilled with all the sections that belong to the selected department:



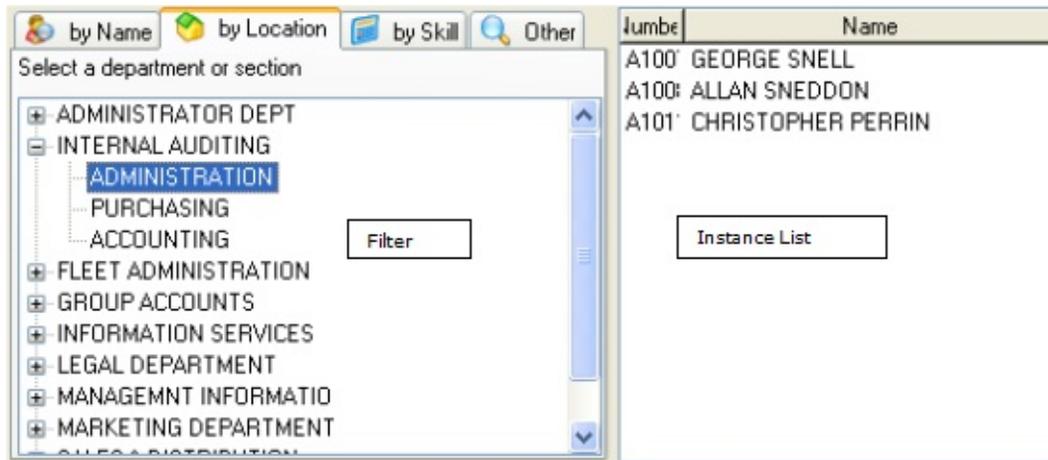
The screenshot shows a software interface with two windows. The left window, titled 'Select Department', contains a list of departments. The right window, titled 'Department', contains a table with columns for Department, Code, Section, Phone, and Zip. The 'GROUP ACCOUNTS' department is selected in the list, and the table displays data for this department.

Department	Code	Section	Phone	Zip
GAC	01	ADMINISTR.	958 4593	2048
GAC	02	PURCHASIN	439 3469	2063
GAC	03	ACCOUNTIN	449 3365	2050
GAC	FC	FINANCIAL	9977 1212	2000

The shipped filter DF_FILT4 provides an example of an "Instant" filter.

Drill Down Filters

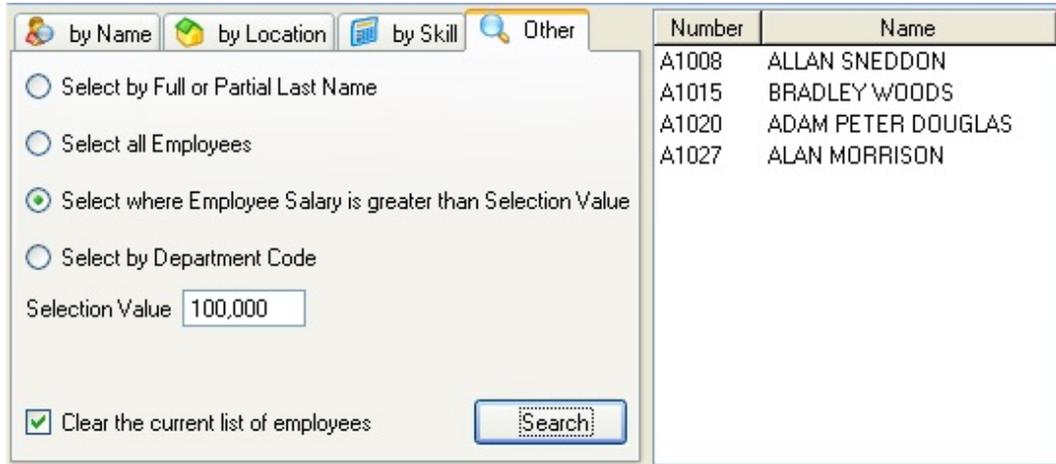
A drill-down filter can, for example, present a tree of departments (at level 1) and their associated sections (at level 2). From the tree you can select a department and then drill down to the sections within the department. As you select departments and sections in the tree the associated employees appear instantly in the instance list:



The shipped filter DF_FILT3 provides an example of a "Drill Down" filter.

Power Filters

You can pack a lot of search power into a single filter by providing multiple search capabilities like this example:



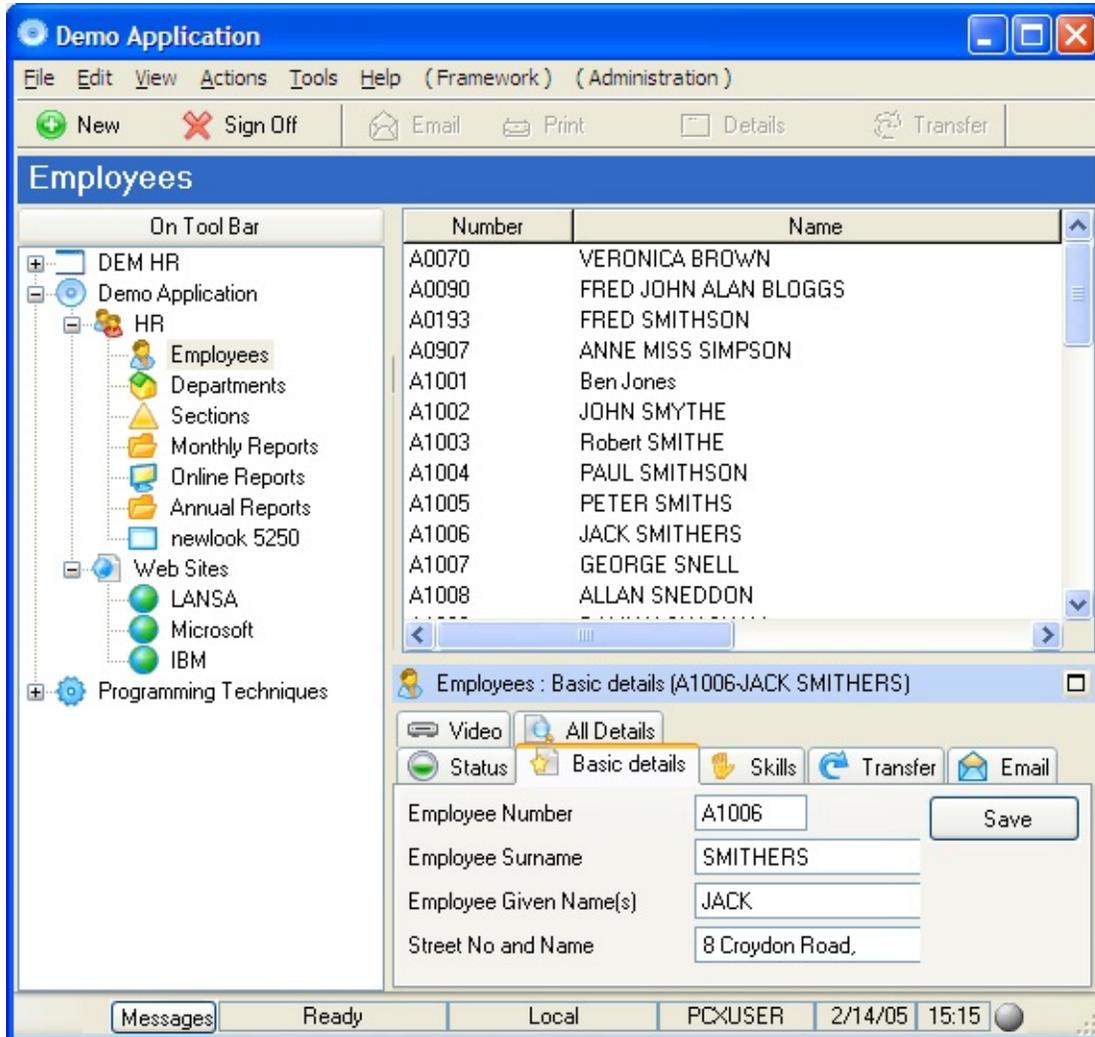
The screenshot shows a software interface with a search filter panel on the left and a table of results on the right. The filter panel has four tabs: "by Name", "by Location", "by Skill", and "Other". The "Other" tab is selected. Inside the "Other" tab, there are four radio button options: "Select by Full or Partial Last Name", "Select all Employees", "Select where Employee Salary is greater than Selection Value" (which is selected), and "Select by Department Code". Below these options is a text input field labeled "Selection Value" containing the number "100,000". At the bottom of the filter panel, there is a checked checkbox labeled "Clear the current list of employees" and a "Search" button.

Number	Name
A1008	ALLAN SNEDDON
A1015	BRADLEY WOODS
A1020	ADAM PETER DOUGLAS
A1027	ALAN MORRISON

Hidden Filter

In situations where you want to completely fill the business object instance list programmatically, the filter has no meaningful interaction with the end-user and can be hidden from view.

Here is an example of a hidden filter that selects all the employees in an HR system and adds them to the instance list:



Notice that the filter is not visible. This filter has a uInitialize routine like this:

```
Mthroutine Name(uInitialize) Options(*Redefine)
```

```
* Do the ancestor thing ....
```

Invoke #Com_Ancestor.uInitialize

* Define this as a hidden filter (you can only ever sensibly have a
* single filter when it's hidden)

Set #Com_Owner avHiddenFilter(TRUE)

* Now fill the instance list with all employee details

Invoke #avListManager.BeginListUpdate

Invoke #avListManager.ClearList

Select Fields(#EmpNo #SurName #GiveName) From_File(PSLMST)

Use BConcat (#GiveName #SurName) #Std_TextL

Invoke Method(#avListManager.AddtoList) Visualid1(#Empno) Visualid2(#St
EndSelect

Invoke #avListManager.EndListUpdate

* Finished

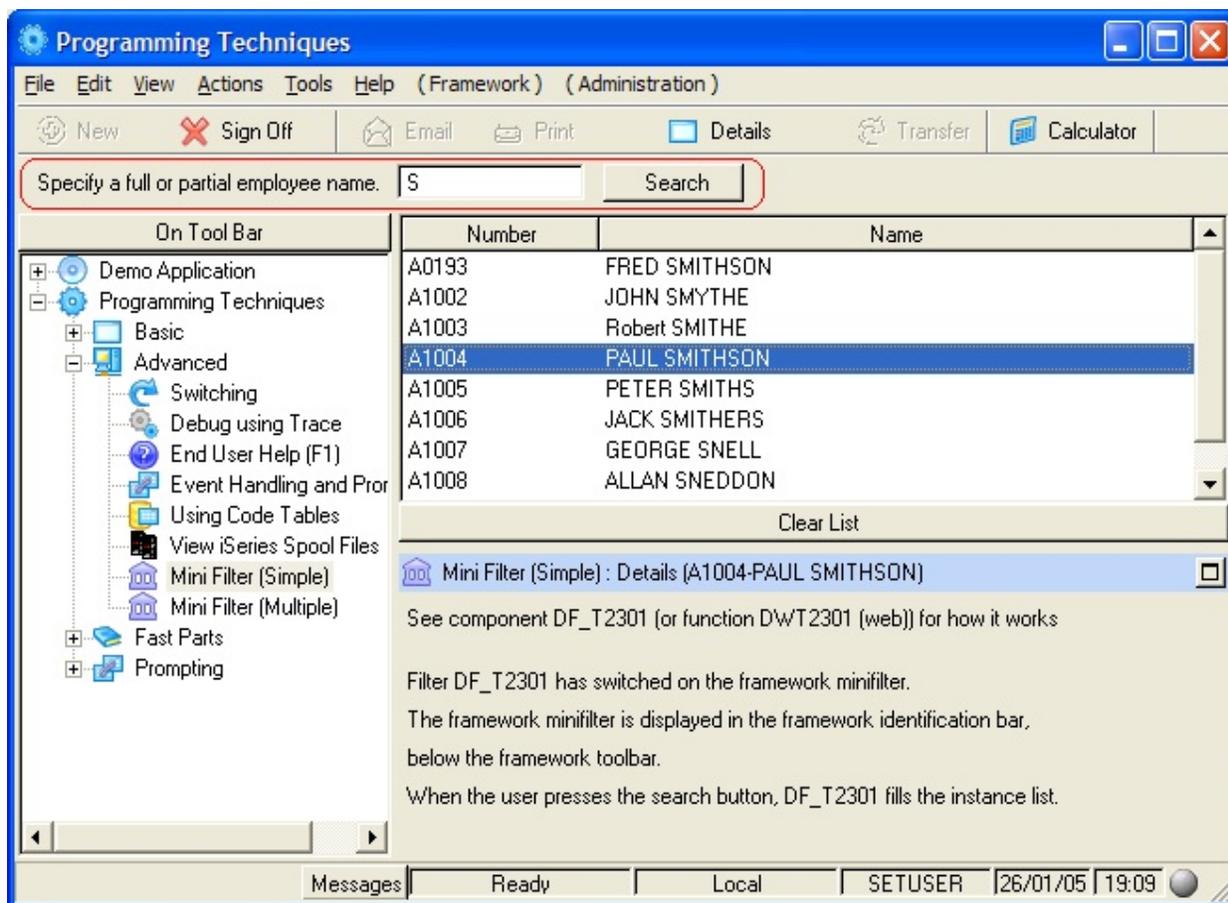
EndRoutine

Mini Filters

[Properties](#) [Events](#) [RDML for a Windows Mini Filter](#) [RDMLX for a WAM Mini Filter](#)

The tutorial [VLF010WIN - Creating a Mini Filter](#) shows how to create a mini filter.

A mini filter very similar to a hidden filter. The difference is that a mini filter still requests a search value from the user, using a field and search button that exist permanently on the Framework identification bar (the field and search button are only visible if a mini filter is being used).



The filter sets up the search field and button on the identification bar. The filter itself is hidden. The user enters a search value into the search field on the identification bar and presses the button on the identification bar. This executes the filter, which gets the value that the user entered, and uses it to do a search and put the result on the instance list.

The filter sets up and communicates with the search field on the identification bar using Framework properties.

Send Properties

Windows:

```
Set #Com_Owner avMiniFilter(TRUE)
set #Com_Owner avMiniFiltFldCap1('Search for')
set #Com_Owner avMiniFiltValue1('<<Initial search value>>')
set #Com_Owner avMiniFiltButtonCap1('Search')
```

Receive Properties

It receives the value in a search field by reading the Framework property for that search field:

```
USE BUILTIN(VF_GET) WITH_ARGS(AVMINIFILTVALUE1 *BLANKS)
```

Windows:

Windows filters receive information as parameters of a method routine. This includes the number of the button that was pressed.

```
Mthroutine Name(uMiniFilterButton) Desc('Handle a mini filter button click')
* Define_Map For(*input) Class(#vf_elnum) Name(#uButton)
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue1) Mandatory(' ')
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue2) Mandatory(' ')
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue3) Mandatory(' ')
```

There can be up to 3 search fields and up to 3 buttons. The search fields are always alpha – if the user enters a numeric value, the RDML of the filter could be coded to convert it to numeric and issue a message if there is a problem.

The routine in the filter that handles a click of a button has access to the values in any of the 3 fields.

If a mini filter is being used, then there should only be one filter for the business object. The reason for having a mini filter is to save space to allow the instance list to be wider, so if there are other filters you should use a normal filter, not a mini filter.

Properties

n on the end of a property name can be 1,2, or 3.

These properties are set in the initialize routine of the RDML of a filter.

Name	Description	Type	Possible Values	MaxLe
avMiniFilter	Indicates the filter is a mini filter	Boolean	TRUE FALSE	5
avMiniFiltValuen	The (alpha) contents of the search field	Character		256
avMiniFiltFieldn	Indicates this field is to be displayed	Boolean	TRUE FALSE	5
avMiniFiltButtonn	Indicates this button is to be displayed	Boolean	TRUE FALSE	5
avMiniFiltFldCapn	The caption that the mini filter field will display	Character		30
avMiniFiltFldCapWidn	The width of the caption	pixels		999
avMiniFiltFldWidn	The displayed width of the caption and field	pixels		999

avMiniFiltButtonCapn	The caption that the mini filter button will display	Character		30
AvMiniFilterPanel	The panel that is to be displayed to manage the mini-filter content. Valid in Windows applications only. When used all other filter properties are not applicable, because they are entirely supplanted by the windows panel specified here. For example you can add combo boxes, drop downs, check boxes, and	#PRIM_PANL	Reference to a panel contained within and owned by the VF_AC007 filter object or *null Any nominated panel must be: <ul style="list-style-type: none"> Owned by the filter itself, not by an ancestor. Not be the filter itself (#Com_Owner). Not layout managed by the filter. 	N/A

do instant
editing on
the panel.

Events

Name	Description
(Windows)	This method routine in the filter is executed when the user clicks on any search button.
Mthroutine	
Name(uMiniFilterButton)	This method routine in the filter is coded to a) work out which button was pressed b) get the value that the user put in the search field, and use it to execute the search logic and load the instance list with the result.

Code Examples

[RDML for a Windows Mini Filter](#)

[RDMLX for a WAM Mini Filter](#)

RDML for a Windows Mini Filter

```
=====
* Type          : FILTER
* Ancestor      : VF_AC007
* Application    : Programming Techniques (Advanced)
* Business Object : Mini Filter (Simple)
* Filter        : Mini Filter (Simple)
* Note: This should be the only filter for this business object
* =====

Begin_Com Role(*EXTENDS #VF_AC007) Height(138) Width(303)
* =====
* Component definitions
* =====

* Nothing is displayed by this filter.
* The Framework mini filter supplies the search button and the search field
* This filter communicates with the mini filter on the Framework identification
* using properties like avMiniFiltField1, and using the redefined method routine
* called uMiniFilterButton.

* =====
* Method Routine Redefines
* =====

*
* Initialize
*

Mthroutine Name(uInitialize) Options(*REDEFINE)

* Make the mini filter visible for this business object
* Note: This should be the only filter for this business object
set #Com_Owner avMiniFilter(TRUE)

* Enable one search field and one search button on the mini filter
* Give them captions, and widths if required
```

```

set #Com_Owner avMiniFiltField1(TRUE)
* set the total field width (including caption) to 300
set #Com_Owner avMiniFiltFldWid1(300)

set #Com_Owner avMiniFiltFldCap1(*MTXTDF_SURNAME)
* set the caption width to 200
set #Com_Owner avMiniFiltFldCapWid1(200)

set #Com_Owner avMiniFiltButton1(TRUE)
set #Com_Owner avMiniFiltButtonCap1(*MTXTDF_SEARCH)
* set #Com_Owner avMiniFiltButtonWid1(200)

* You could also pass an initial search value using:
* set #Com_Owner avMiniFiltValue1('my initial search value')

endroutine

*
* Perform Search instruction from the mini search panel
*

Mthroutine Name(uMiniFilterButton) Desc('Handle a mini filter button click')
* Define_Map For(*input) Class(#vf_elnum) Name(#uButton)
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue1) Mandatory(' ')
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue2) Mandatory(' ')
* Define_Map For(*input) Class(#vf_elurl) Name(#uValue3) Mandatory(' ')

Define #Button *dec 1 0

* Determine which mini filter button the user pressed
change #Button #uButton
case #button
when '= 1'

* Get the search value that the user entered into the mini filter
USE BUILTIN(UPPERCASE) WITH_ARGS(#UVALUE1) TO_GET(#SURNAME)

* Do the search and put the results into the instance list

```

```
Invoke Method(#avListManager.BeginListUpdate)
```

```
Invoke Method(#avListManager.ClearList)
```

```
* Select employee records using full or partial surname in surname order and a  
Select Fields(#EMPNO #GIVENAME #SURNAME) From_File(PSLMST2) \
```

```
* add an entry to the instance list on the right of the filter
```

```
Execute Subroutine(ADDLIST) With_Parms(#SURNAME #GIVENAME #EM  
Endselect
```

```
* Indicate end of list update
```

```
Invoke Method(#avListManager.EndListUpdate)
```

```
endcase
```

```
endroutine
```

```
* =====
```

```
* Subroutines
```

```
* =====
```

```
*
```

```
* add an entry to the instance list on the right of the filter
```

```
*
```

```
Subroutine Name(ADDLIST) Parms(#SURNAME #GIVENAME #EMPNO)
```

```
* Build up visual identifier
```

```
Use BConcat (#GiveName #SurName) #FullName
```

```
Invoke Method(#avListManager.AddtoList) Visualid1(#Empno) Visualid2(#Fu
```

```
Endroutine
```

```
End_Com
```

RDMLX for a WAM Mini Filter

```
*
=====
* Description ...: Mini Filter (Simple)
*
=====
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('vlf_layout')

* Standard declares for a filter

Define_Com Class(#vf_sw100) Name(#avFrameworkManager)
Define_Com Class(#vf_aw007) Name(#ThisFilter) Reference(*Dynamic)
Define_Com Class(#vf_lw002) Name(#avListManager)
Reference(*Dynamic)
Define_Com Class(#fp_in001) Name(#FastPart) Reference(*Dynamic)

* VL Framework map fields. DO NOT CHANGE.

Web_Map For(*both) Fields((#VF_FRAMEI *private) (#VF_FRAMEW
*private) (#VF_FRAMES *private) (#VF_ELXTOF *private) (#VF_ELXN01
*private) (#VF_ELXN02 *private) (#VF_ELXN03 *private) (#VF_ELXN04
*private) (#VF_ELXN05 *private) (#VF_ELXN06 *private) (#VF_ELXN07
*private) (#VF_ELXN08 *private) (#VF_ELXN09 *private) (#VF_ELXN10
*private) (#VF_ELXA01 *private) (#VF_ELXA02 *private) (#VF_ELXA03
*private) (#VF_ELXA04 *private) (#VF_ELXA05 *private) (#VF_ELXA06
*private) (#VF_ELXA07 *private) (#VF_ELXA08 *private) (#VF_ELXA09
*private) (#VF_ELXA10 *private) (#VF_ELMETA *private))

* Standard webroutine used by all VLF WAM filters and command handlers.
DO NOT CHANGE the Web routine name. Use this routine to register WAM
events.

Webroutine Name(UHandleEvent)

* Standard WAM initialisation.
```

* NOTE: THE EVENTS FOR MINI FILTERS ARE PRE REGISTERED IN THE FRAMEWORK. THEREFORE, YOU DON'T NEED TO REGISTER THEM. THEY ARE:

- * uWAMEvent_21 --> mini filter button ONE was clicked
- * uWAMEvent_22 --> mini filter button TWO was clicked
- * uWAMEvent_23 --> mini filter button THREE was clicked

```
#avFrameworkManager.avInitializeWAM Type(FILTER)
```

```
Invoker(#com_owner) Listmanager(#avListManager)
```

```
Filtermanager(#ThisFilter) Fastpartmanager(#FastPart)
```

```
#avFrameworkManager.avHandleWAMEvent Anchorblock(#vf_framew)
```

```
Event(#vf_event) Designmode(#vf_framed) Skin(#VF_Frames)
```

```
Metatag(#vf_elmeta) Tof(#vf_elxtof) N01(#vf_elxn01) N02(#vf_elxn02)
```

```
N03(#vf_elxn03) N04(#vf_elxn04) N05(#vf_elxn05) N06(#vf_elxn06)
```

```
N07(#vf_elxn07) N08(#vf_elxn08) N09(#vf_elxn09) N10(#vf_elxn10)
```

```
A01(#vf_elxA01) A02(#vf_elxA02) A03(#vf_elxA03) A04(#vf_elxA04)
```

```
A05(#vf_elxA05) A06(#vf_elxA06) A07(#vf_elxA07) A08(#vf_elxA08)
```

```
A09(#vf_elxA09) A10(#vf_elxA10) Ssiname(#VF_FRAMEI)
```

Endroutine

- * Handle initialization of filter

```
Evtroutine Handling(#avFrameworkManager.uexecute)
```

```
Options(*noclearmessages *noclearerrors)
```

- * Activate this Filter as a mini Filter

- * This filter will use a search field and search button on the framework identification bar (below the toolbar)

- * to interact with the user. This filter will not be displayed.

- * Note: This should be the only filter for this business object.

- * Enable one search field and one search button on the mini filter. Give them captions, and widths if required.

```
Set Com(#thisfilter) Avminifilter(true) Avminifiltfield1(true)
```

```
Avminifiltfldcapwid1(250) Avminifiltfldcap1(*MTXTDF_SURNAME)
Avminifiltfldwid1(350) Avminifiltbutton1(true)
Avminifiltbuttoncap1(*MTXTDF_SEARCH) Avminifiltbuttonwid1(150)
```

```
Endroutine
```

```
Evtoutine Handling(#avFrameworkManager.uWAMEvent_21)
Options(*noclearmessages *noclearerrors) Value1(#Val1)
```

```
* The value that the user entered on the search field on the framework
identification bar is provided in the Value1 parameter of this event routine.
#Surname := #Val1.UpperCase
```

```
* Start instance list update. Clear the instance list
#avListManager.BeginListUpdate
#avListManager.ClearList
```

```
* Fill the instance list based on the search value specified by the user.
Select Fields(#SURNAME #GIVENAME #EMPNO #PHONEHME
#ADDRESS1 #POSTCODE) From_File(PSLMST2)
With_Key(#SURNAME) Generic(*YES)
Execute Subroutine(ADDLIST)
Endselect
```

```
* Instance list updating has been completed
#avListManager.EndListUpdate
```

```
Endroutine
```

```
* =====
* SUBROUTINE: Add an entry to Instance List
* =====
```

```
Subroutine Name(ADDLIST)
```

```
* Build up visual identifier
```

```
#Fullname := #Givename + " " + #Surname
```

```
* Add to instance list using the user defined
* identification protocol:
```

```
* =====
* Visual Identification
* =====
* VisualID1 = Employee Number
* VisualID2 = Concatenation of first name (#GiveName)
* and last name (#SurName)
* =====
* Programmatic Identification
* =====
* AKey1 = Employee number
* =====
* Additional Columns for instance list manager
* =====
* AColumn1 = Phone (Home)
* AColumn2 = Address Line 1
* NColumn1 = Post Code (Zip)
#avListManager.addToList Visualid1(#Empno) Visualid2(#Fullname)
Akey1(#Empno) Acolumn1(#Phonehme) Acolumn2(#Address1)
Ncolumn1(#Postcode)

Endroutine

End_Com
```

Web Programming

The Visual LANSА Framework allows you to create filters or command handlers that may be executed using a Web Browser interface. It does this by using LANSА for the Web's Web Application Modules (WAMs).

Programming WAM applications is different to programming 5250 based LANSА for i applications or Windows Visual LANSА applications.

Another option you have is to create [Framework-AJAX Applications](#).

WAM Programming

Stateless Programming

UB_XXXXX User Buttons

Maintaining State in WAM Filters and Command Handlers

Using LANSAs Weblets with Framework WAMs

Rules for WAM Filters and Command Handlers

Stateless Programming

The fundamental difference between Web programming and LANSAs for i or Visual LANSAs programming has to do with understanding stateless programs. Programs are stateless when they do not maintain any state between their interactions with the browser user (or anything else out there on the Internet). This means that the variables defined in the program cannot be used to remember things between interactions. Other more subtle states cannot be kept intact either, such as the location of a file cursor, an open file connection or an allocated resource.

To illustrate the impact of stateless programming to consider this simple program specification:

- Display a field called #NUMBER that starts with value 0. Every time the user presses enter increment #NUMBER and display the new result.

In 5250 programs (LANSAs/RPG) or in a Windows programs (Visual LANSAs/Visual Basic) the logic would be:

```
Define #NUMBER with a default value of zero
Change #NUMBER to #NUMBER + 1
Display the incremented #NUMBER value
```

In stateless programs #NUMBER (and in fact your whole program and all its state) **ceases to exist** between interactions with the user. Therefore the variable cannot be used to remember its previous value between interactions with the user.

The change to your programming approach is easy enough. You simply need to do this:

```
Define #NUMBER
Restore the current value (i.e. state) of #NUMBER
Change #NUMBER to #NUMBER + 1
Save the new value (i.e. state) of #NUMBER
Display the new #NUMBER value (at which point your program ceases to exist)
```

LANSAs for the Web and the Framework both provide facilities to save and restore state for you (more on that later). Maintaining state is easy, but you need to understand that it is happening and how it impacts your applications.

UB_XXXX User Buttons

The Framework is shipped with a set of fields and weblets named UB_XXXX. These are a collection of weblets which you can use in WAM based filters and command handlers.

There is a UB_XXXX weblet for each UB_XXXX field.

In addition, each UB_XXXX field shipped has a visualization attribute of the corresponding weblet.

The code in the UB_XXXX weblet draws the UB_XXXX button on the form.

How are they used?

You can place the user buttons on filters and command handlers and use them to trigger functions, for example Save. There are button weblets for the most common uses, for example OK, Save and Delete, and spare buttons you can format for your own particular uses, for example Show Details or Calculate.

How can the captions be changed?

The buttons not already assigned to a particular purpose can be modified to display any caption. The easiest way of doing this is to override the field definition:

```
OVERRIDE FIELD(#UB_PUSHB1) DEFAULT("Show Details")
```

Or if the application is multilingual then a multilingual text variable would be used:

```
OVERRIDE FIELD(#UB_PUSHB1) DEFAULT(*MTXTDF_SHOWDETAIL
```

How are the buttons placed on a web form?

You place the user buttons on web forms by including them in a **Web_Map For(*both)** statement of the command handler or filter.

Once the UB_XXXX field is used in a Web_Map, you may choose to generate the XSL for the WAM. In such case, the field appears somewhere on the page as a button.

However, you would normally want to paint the form yourself to have more control over where all the form elements appear. In such case you would just drag the UB_XXXX weblet and drop it in the desired place on the form.

The tabbing order of buttons may be controlled

Use the TabIndex property of the avSetButton method.

For example:

```
Invoke Method(#ThisHandler.avSetButton) ButtonName(UB_SAVE)
TabIndex('1')
Invoke Method(#ThisHandler.avSetButton) ButtonName(UB_SAVE)
TabIndex('2')
```

User buttons can be hidden or disabled

User buttons can be added to a web form and then hidden or disabled if they are not relevant.

To hide or disable a button use the avSetButton method of the command handler or filter with the Visible or Enabled keywords. For example, to hide the Save button in a Command Handler defined as

```
Define_Com Class(#vf_aw010) Name(#ThisHandler) Reference(*Dynamic)
```

you would

```
Invoke Method(#ThisHandler.avSetButton) Buttonname(UB_SAVE) Visible
```

To disable it:

```
Invoke Method(#ThisHandler.avSetButton) Buttonname(UB_SAVE) Enabled
```

How to detect when a button is pushed?

You must register a Wam event in the uHandleEvent web routine where the Named keyword has normally the format

<field name>.CLICK

For example, to register a Wam event which fires when the user clicks on the Save button:

```
#avFrameworkManager.avRegisterEvent Named(UB_SAVE.CLICK) Signal:
```

You would then require an event routine to handle that event:

```
Evtroutine Handling(#avFrameworkManager.uWAMEvent_n) Options(*nocle:
```

A simple example of a wam command handler that uses a user button

```
Begin_Com Role(*EXTENDS #PRIM_WAM) Layoutweblet('vlf_layout')
Define_Com Class(#vf_sw100) Name(#avFrameworkManager)
Define_Com Class(#vf_aw010) Name(#ThisHandler) Reference(*Dynamic)
Define_Com Class(#vf_lw002) Name(#avListManager) Reference(*Dynamic)
Define_Com Class(#fp_in001) Name(#FastPart) Reference(*Dynamic)
```

* VL Framework map fields. DO NOT CHANGE.

```
Web_Map For(*both) Fields((#VF_FRAMEI *private) (#VF_FRAMEW *priv
*PRIVATE) (#VF_ELMETA *PRIVATE))
```

* Map fields used in this form.

```
Web_Map For(*both) Fields((#empno *ouput) #surname #givename #salary (#
```

```
Webroutine Name(UHandleEvent)
```

* Register the event that will execute when clicking on the Save button in this :

```
Invoke Method(#avFrameworkManager.avRegisterEvent) Named(UB_SAVE.(
```

* Standard WAM initialisation.

```
Invoke Method(#avFrameworkManager.avInitializeWAM) Type(COMMAND)
Invoke Method(#avFrameworkManager.avHandleWAMEvent) Anchorblock(#
Endroutine
```

* Initialize the command handler

```
Evtoutine Handling(#avFrameworkManager.uInitialize) Options(*noclearmes
```

* Get the current employee number from the instance list

```
Invoke Method(#avListManager.getCurrentInstance) Akey1(#Empno)
```

```
Endroutine
```

* Handle execution of the command handler

```
Evtoutine Handling(#avFrameworkManager.uexecute) Options(*noclearmess
```

```
Fetch Fields(#SURNAME #GIVENAME #SALARY) From_File(PSLMST) W
```

```
Endroutine
```

* Handle click of the save button

```
EvtRoutine Handling(#avFrameworkManager.uWAMEvent_1) Options(*nocle:
```

```
Update Fields(#GIVENAME #SURNAME #SALARY) In_File(PSLMST) Wit
```

```
Endroutine
```

```
End_Com
```

Maintaining State in WAM Filters and Command Handlers

You may need to maintain state in filters and command handlers. There are various ways to do this:

[Use the WAM capabilities](#)

[Use the Framework's Virtual Clipboard](#)

Use the WAM capabilities

Visual LANSA Framework filters and command handlers have normally one Webroutine. Thus to maintain the state of fields in Framework filters and command handlers is quite simple. You must include the fields in a Web_Map. You would normally define a Web_Map for both input and output.

This means that the simple incremental program defined earlier as:

- Define #NUMBER with an initial value of zero.
- Change #NUMBER to #NUMBER + 1.
- Display the incremented #NUMBER value.

Would function correctly when coded like this:

```
Function Options(*Direct)
Define #Number *dec 7 0 default(0)
Web_Map For(*both) Fields(#Number)

Webroutine Name(UHandleEvent)
etc
Endroutine
```

Use the Framework's Virtual Clipboard

The Framework extends the LANSAs for the web state model by providing a facility called the virtual clipboard.

Using the virtual clipboard you might code the function like this:

```
Define #Number *dec 7 0 default(0)
Web_Map For(*both) Fields(#Number)
```

To save the #Number value:

```
#avFrameworkManager.avSaveValue Withid1(*Component) Withid2(Save_Nu
```

To restore the #Number value:

```
#avFrameworkManager.avRestoreValue Withid1(*Component) Withid2(Save_
```

The virtual clipboard also allows you to dynamically define complex lists (and even lists within lists). It also allows information to be easily passed between filters and command handlers.

The shipped Programming Techniques application contains several examples of using the Virtual Clipboard facility.

Using LANSA Weblets with Framework WAMs

In Visual LANSA Framework WAM applications you can use LANSA Weblets.

When using LANSA Weblets you need to be aware of the following:

- When LANSA Weblets submit requests to the web server, they need to be identified to your WAM filter or command handler as an 'event'.
- The event's identifier allows your WAM filter or command handler to determine which Weblet made the request and act accordingly (for example Which button was clicked? Ok, Save or Cancel?).
- The identification of an event involves using a special property named `VF_WAMEVENT`.
- The value assigned to it identifies the event to your filter or command handler.

To use LANSA Weblets in VLF WAMs follow these steps:

1. Add the LANSA weblet to the VLF filter or command handler WAM in the WAM editor of the LANSA IDE.
2. On the Weblet's details tab set the property `VF_WAMEVENT` to the identifier of the event, e.g. 'BUTTON.CLICK'. Note that the value must be enclosed in single quotes.
3. Set the `on_click_wamname` property of the weblet to the name of the VLF filter or command handler WAM, e.g. 'DM_T2901', and the `on_click_wrname` property to 'uHandleEvent'. The correct values for these properties will be available from dropdowns on the property fields.
4. Register an event in the VLF WAM with the same event identifier as used in the `reentryvalue` property of the LANSA weblet.
5. Create a corresponding event handler in the VLF WAM to perform the required processing.

Details

Properties

<xsl:call-template>

Name	Value
on_click_wname	UHandleEvent
protocol	
show_in_new_window	False
target_window_name	<xsl:if xmlns:xsl="http://v
disabled	False
title	
text_class	"
presubmit_js	
confirm	False
confirmText	
tab_index	
default_button	
vf_wamevent	BUTTONV2.CLICK

entryvalue property of the LANSAs weblet.
cessing.

WAM filter or command handler as an 'event'. The event's
equest and act accordingly (eg: The LANSAs Button).



Webroutine Name(UHandleEvent)

↓

* Register the event that will execute when clicking on the Button in this command
Invoke Method(#avFrameworkManager.avRegisterEvent) **Named(BUTTONV2.CLICK)** Signalaswamevent(1)

* Standard WAM initialisation.

```
Invoke Method(#avFrameworkManager.avInitializeWAM) Type(COMMAND) Invoker(#com_owner)
  Listmanager(#avListManager) Handlermanager(#ThisHandler) Fastpartmanager(#FastPart)
Invoke Method(#avFrameworkManager.avHandleWAMEvent) Anchorblock(#vf_framework) Event(#vf_event)
  Designmode(#vf_framed) Skin(#VF_Frames) Metatag(#vf_elmeta) Tof(#vf_elxtof)
  N01(#vf_elxn01) N02(#vf_elxn02) N03(#vf_elxn03) N04(#vf_elxn04) N05(#vf_elxn05)
  N06(#vf_elxn06) N07(#vf_elxn07) N08(#vf_elxn08) N09(#vf_elxn09) N10(#vf_elxn10)
  A01(#vf_elxA01) A02(#vf_elxA02) A03(#vf_elxA03) A04(#vf_elxA04) A05(#vf_elxA05)
  A06(#vf_elxA06) A07(#vf_elxA07) A08(#vf_elxA08) A09(#vf_elxA09) A10(#vf_elxA10)
  Ssiname(#VF_FRAMEI)
```

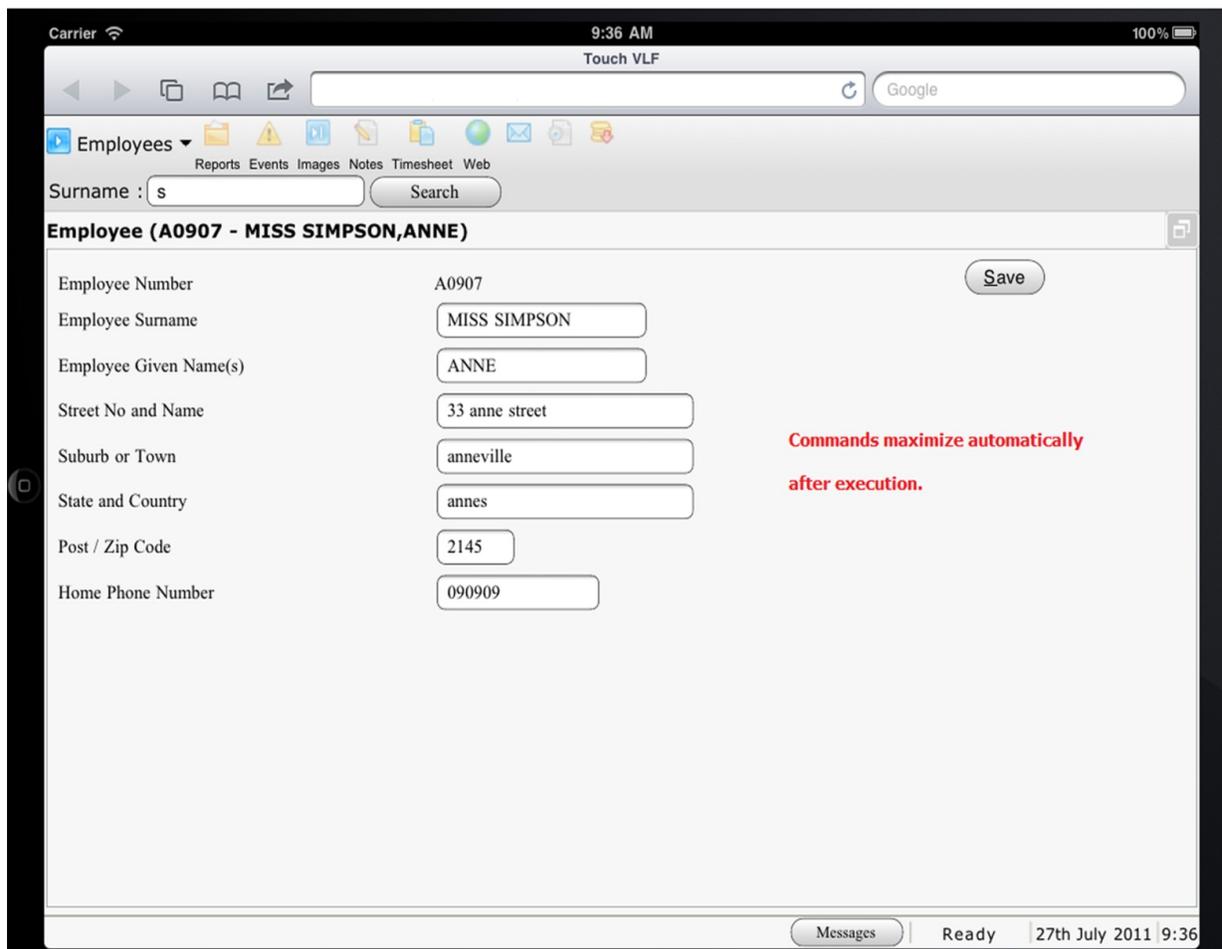
Rules for WAM Filters and Command Handlers

To make sure your filter or command handler meets the rules you should use the Program Code Assistant.

Touch Device Considerations

Framework web applications can be run on iPads and Android touch devices. User interaction in touch devices is based on touch interface. To enable touch friendly functionality in the Framework, use the web startup URL parameter TOUCH=Y .

If this parameter has not been specified or its value is any other than Y, the Framework makes a guess as to whether it is being run on a touch device.



There is also a URL parameter ZOOM= which can be used to change the default CCS zoom value. For more information see [Web Application Start Options](#).

When Designing Applications for Touch Devices

- Use mini filters whenever possible to leave more space for the instance list and commands.

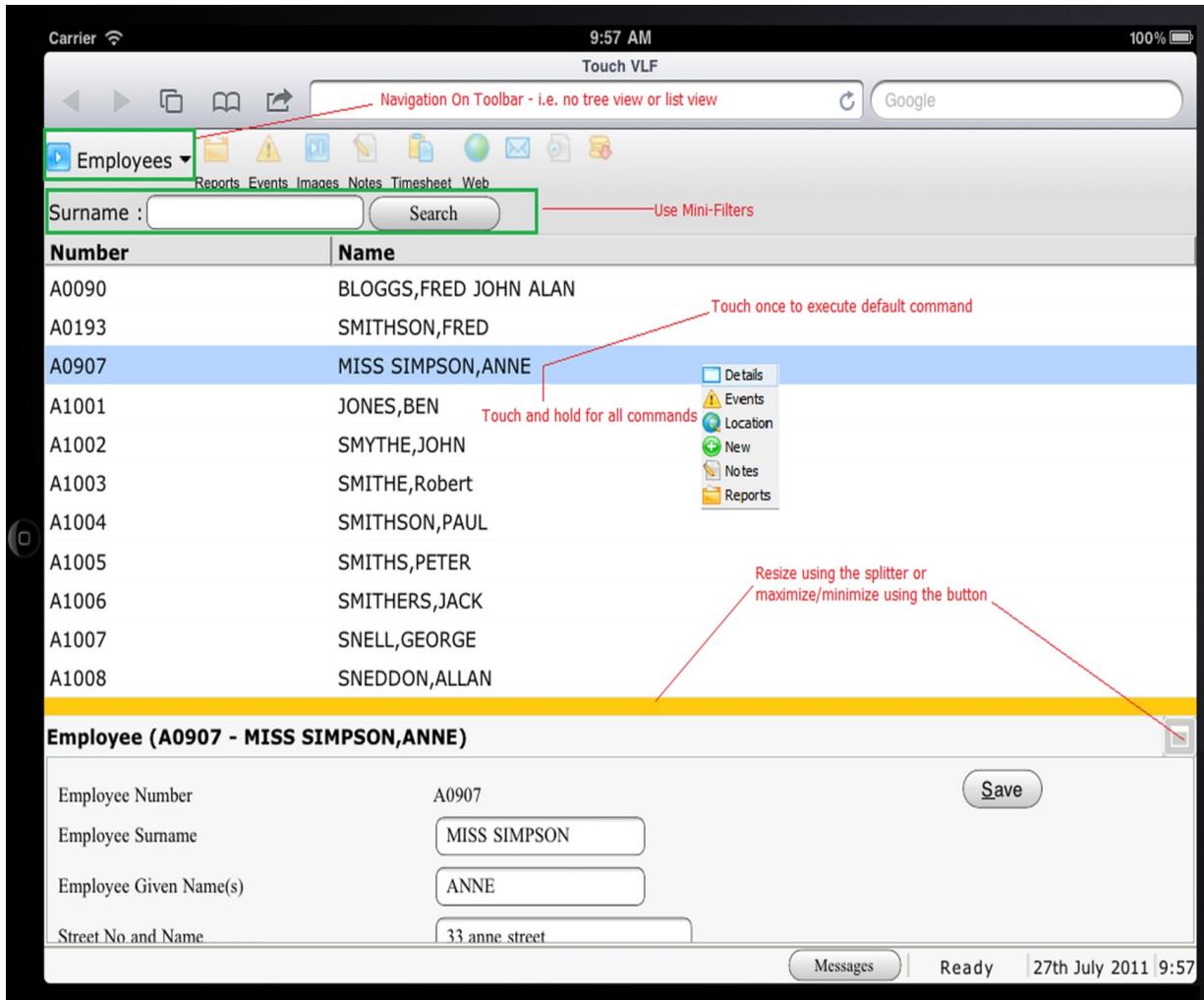
- Avoid tree views in instance lists.
- Note that when the VLF is running on touch devices, the Log off Inactivity Timeout and Log on Inactivity Timeout options are disabled. You should use device specific timeout settings to control access to the device.
- Avoid using command handlers that are displayed in their own windows. Command handlers should always be displayed in one of the tabs.

When Executing A Framework Application on a Touch Device

- In touch devices the Framework will always appear as if the following properties had been set (switching views is not allowed):



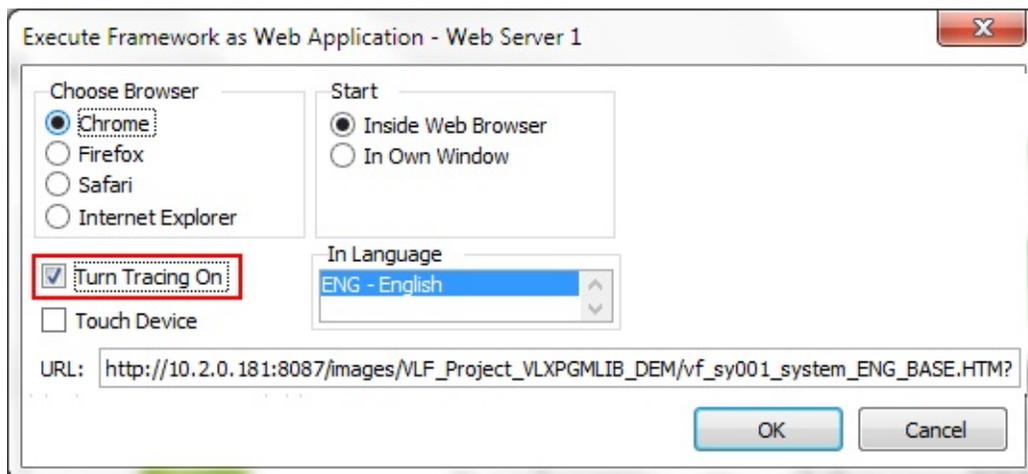
- The splitter bars are wider to make it easier to drag them.
- To minimize the need to resize, commands are automatically maximized when they finish execution.
- One touch on an instance list entry will execute the default command.
- To bring up the context menu, touch and hold for longer than a second.



Also see [Tracing on iPads](#) and [Tracing on Android Touch Devices](#).

Tracing Web Applications

Tracing in Web browser applications can be turned on by checking the Turn Tracing On option when launching your application on the web: You can also add ?Trace=Y to end of any Framework URL to initiate tracing:



To trace values when a WAM first starts use a #avFrameworkManager.uInitialize event handler. Similarly, to trace values as a WAM closes use a #avFrameworkManager.uTerminate event handler:

Trace values when WAM starts:

```
EvtRoutine Handling(#avFrameworkManager.uInitialize)
Options(*noclearmessages *noclearerrors)
```

```
Invoke #AVFRAMEWORKMANAGER.avRecordTrace
Component(#com_owner) Event('WAM Filter is initialising')
Endroutine
```

Trace values when WAM ends:

```
EvtRoutine Handling(#avFrameworkManager.uTerminate)
Options(*noclearmessages *noclearerrors)
```

```
Invoke #AVFRAMEWORKMANAGER.avRecordTrace
Component(#com_owner) Event('WAM Filter is terminating')
Endroutine
```

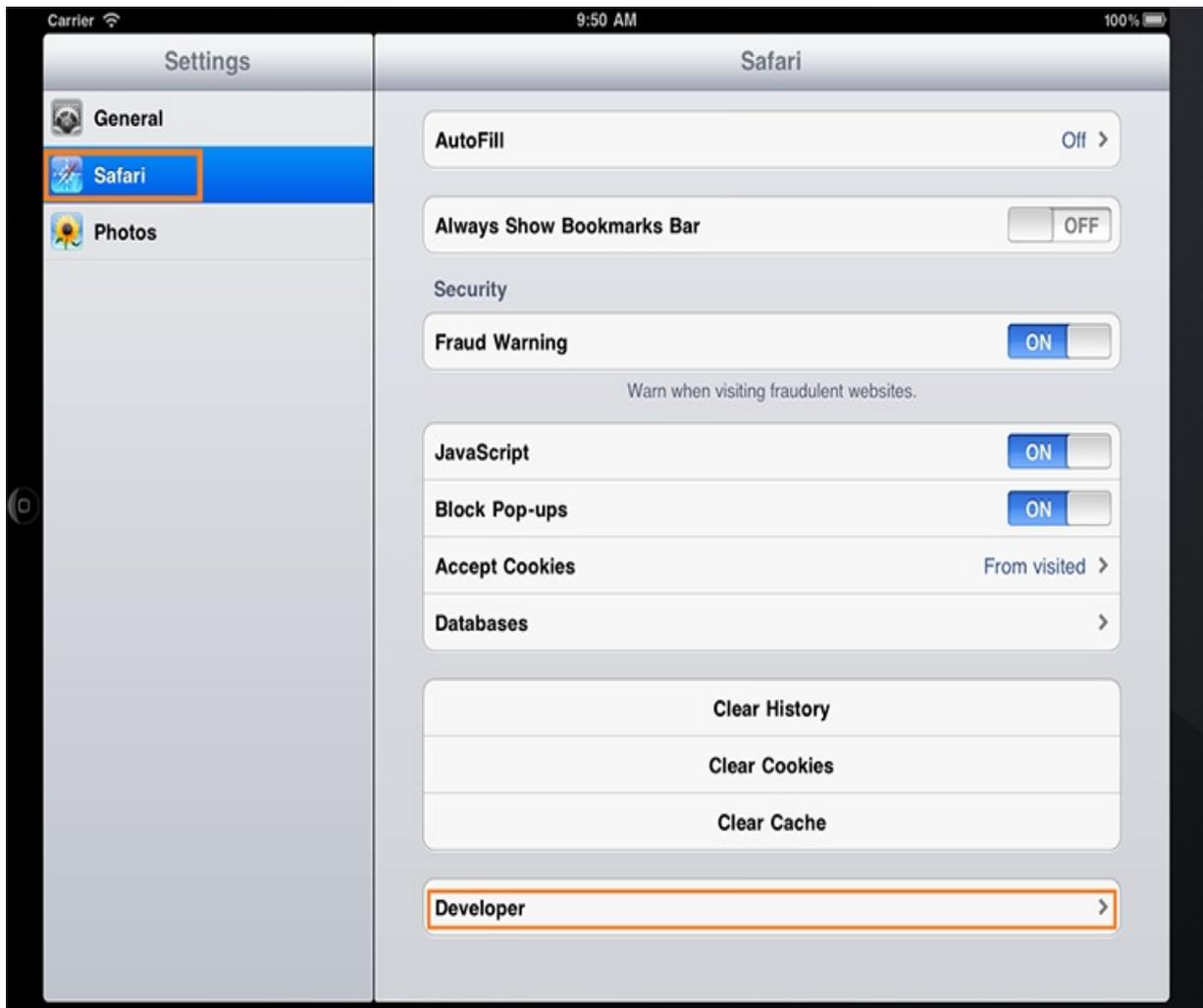
Also see:

Tracing on iPads

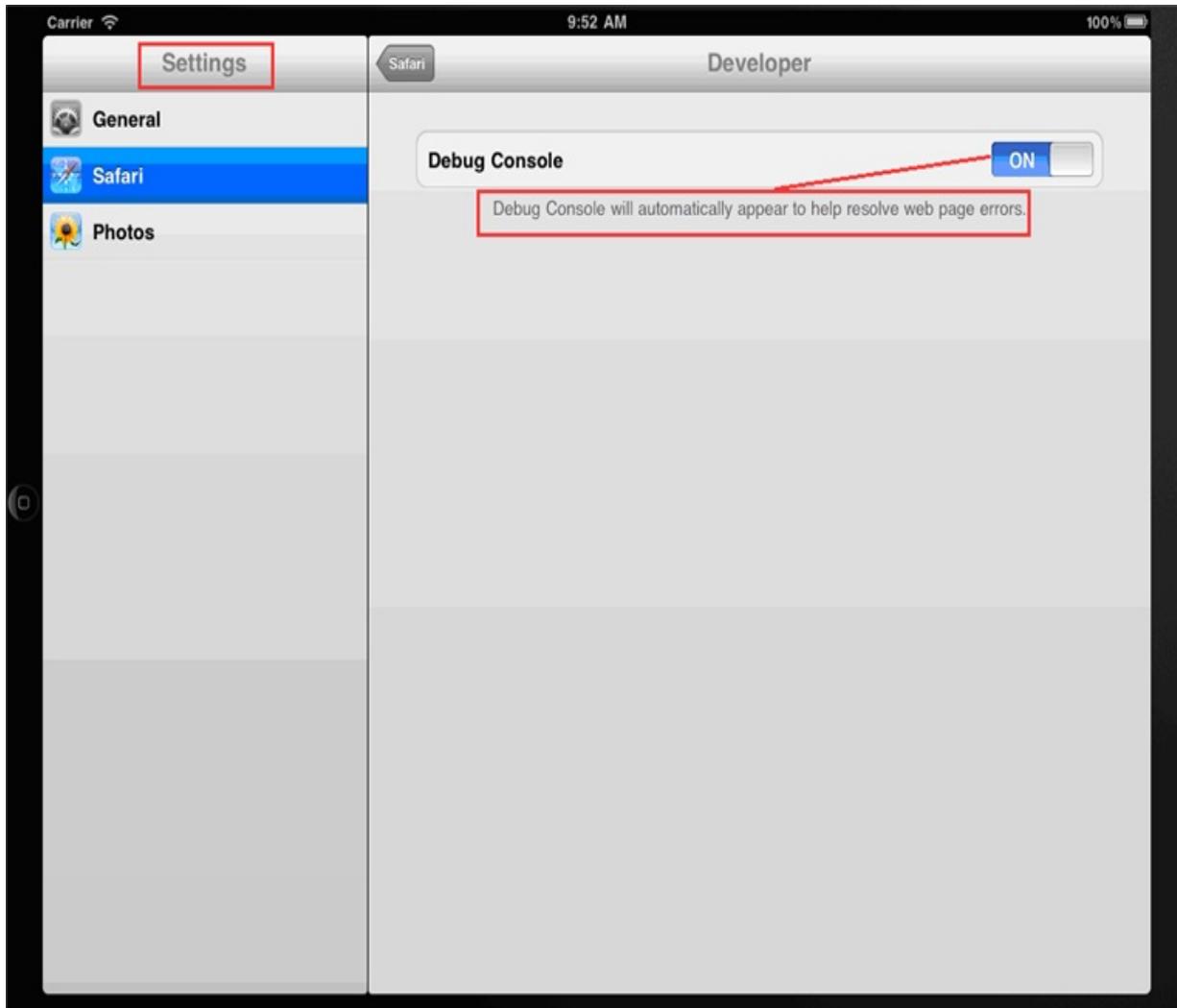
Tracing on Android Touch Devices

Tracing on iPads

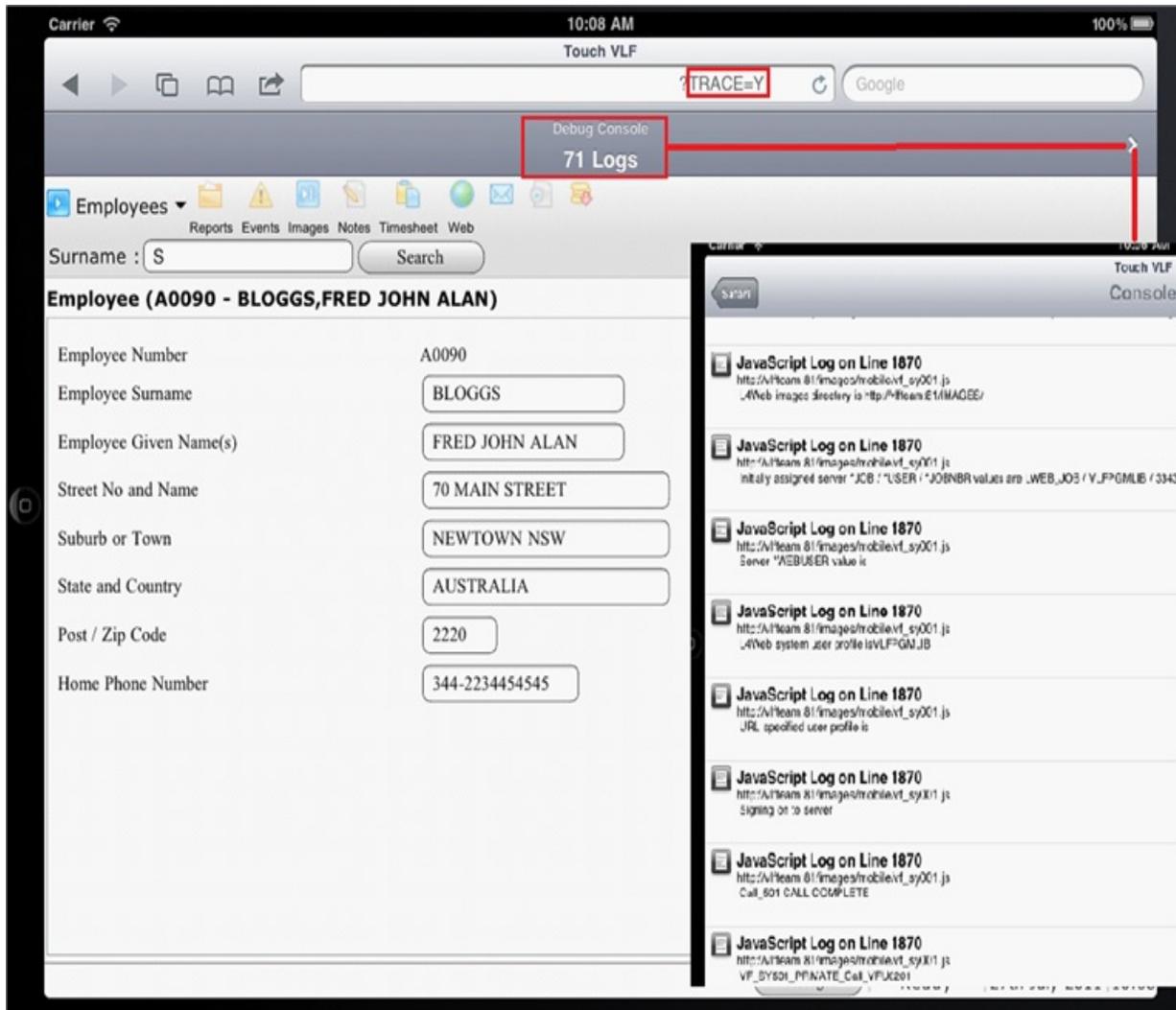
1. In the Settings panel touch Safari and then Developer.



2. In the Developer panel turn the Debug Console on:



3. When you now go back to Safari you can see a small Debug Console area. If there are logs to show, touching the arrow on the right will bring up the debug console like this:



Javascript errors also show up in the Debug Console.

Tracing on Android Touch Devices

Tracing on Android touch devices is not as straightforward as on iPads because there is no browser inbuilt console.

There are a few tools to remote debug Webkit browsers. Here we are only going to cover the Android Debug Bridge – adb – which lets you output trace statements to a dos command prompt. To use adb you require:

- The platform tools included in the Android SDK package: <http://developer.android.com/sdk/index.html>. Note: the entire package is quite big, it should be enough to just get the platform tools.
- A USB connection to a PC or laptop. Check that your device has USB Debugging enabled: Settings – Applications – Development – USB debugging

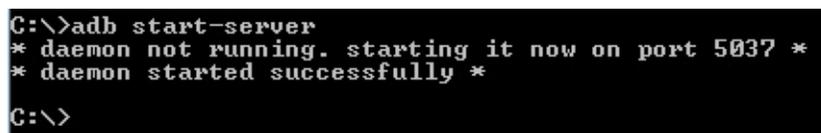
To start using adb:

1. Connect the device to the computer and open a DOS prompt and change directory to the platform tools.
2. Type this command and press Enter:



```
Administrator: Command Prompt
C:\>adb kill-server
```

3. Type this command and press Enter:



```
C:\>adb start-server
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
C:\>
```

4. Type this command and press Enter:

```
Administrator: Command Prompt
C:\>adb devices
List of devices attached
C:\>
```

If no devices are listed it probably means that the proper driver is missing. Otherwise you should see something like this:

```
C:\>adb devices
List of devices attached
2804180437f5317 device
C:\>
```

5. To start tracing, first clear the log:

```
C:\>adb logcat -c
C:\>
```

6. Next you can issue this command which will filter the browser messages:

```
Administrator: Command Prompt - adb logcat browser:I *:S
C:\>adb logcat browser:I *:S
----- beginning of /dev/log/system
----- beginning of /dev/log/main
```

From here on any trace messages will come out on the DOS window:

```
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=ADMINISTRATOR DEPT,VisualID2=ADM,BusinessObj  
e=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=INTERNAL AUDITING,VisualID2=AUD,BusinessObj  
e=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=FLEET ADMINISTRATION,VisualID2=FLT,BusinessO  
bj=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=GROUP ACCOUNTS,VisualID2=GAC,BusinessObject  
ID=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.181:81/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=INFORMATION SERVICES,VisualID2=INF,BusinessO  
bj=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=LEGAL DEPARTMENT,VisualID2=LEG,BusinessObj  
e=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=MANAGEMNT INFORMATIO,VisualID2=MIS,BusinessO  
bj=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=MARKETING DEPARTMENT,VisualID2=MKT,BusinessO  
bj=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=SALES & DISTRIBUTION,VisualID2=SD,BusinessOb  
j=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870  
/browser ( 2288): Console: DWFILT8 Add to instance list executed. VisualID1=TRAVEL DEPARTMENT,VisualID2=TRVL,BusinessOb  
j=DEM_ORG,SubType=,RowColor CSS= http://10.2.0.101/images/mobile/vf_sy001.js:1870
```

For more information about the adb interface see:

<http://developer.android.com/guide/developing/tools/adb.html>.

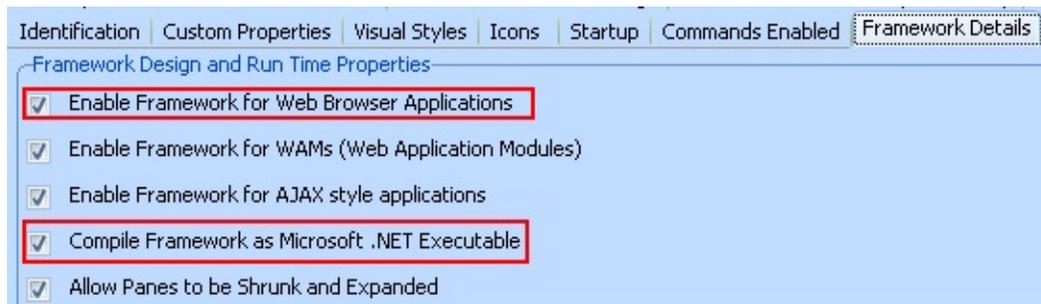
VLF.NET Applications

Any Framework Web Browser application can be compiled as a .NET executable. The VLF.NET feature supports all existing and new WAM filters and command handlers and all RAMP screens and scripting.

How is the VLF.NET feature activated?

To enable the VLF.NET feature:

- Open your Framework as a designer. Use the (Framework) -> (Properties...) menu options. Display the Framework Details tab. Ensure that the Enable Framework for Web Browser Applications option is checked.
- Check the Compile Framework as Microsoft .NET Executable option.



When you next save the Framework with these options selected, the Framework is compiled as a .NET executable.

Are there any circumstances under which VLF.NET feature cannot be used?

Yes, if the users are behind a proxy server that requires them to enter their username and password, the VLF.NET feature cannot be used. This is a limitation of .NET ClickOnce deployment technology which does not currently support downloading of files through a proxy server that requires non-Windows integrated authentication.

Microsoft's .NET Framework

To use VLF.NET both developers and end-users need to have Microsoft's .NET Framework 2.0 (or later) and .NET Framework 2.0 Service Pack 1 installed.

If .NET Framework 3.5 has been installed, .NET Framework 2.0 Service Pack 1 does not need to be installed separately.

Some problems have been reported when installing .NET Framework 2.0 Service Pack 1 on Vista, in which case .NET Framework 3.5 will be have to be installed.

.NET 2.0 Software Development Kit (SDK) is not required.

Deployment

For step-by-step instructions of how to deploy a VLF.NET application, see [Check List/Planning Sheet WEB-NET](#).

Also see [VLF.NET Manual Deployment – An Alternative Way of Deploying VLF.NET](#)

When you save the Framework with the VLF.NET feature activated, additional .NET files are created in addition to the normal VLF.WEB HTML and JS files.

If your Framework was stored in an XML file named My_Framework.XML and enabled for languages ENG and FRA, these additional .NET files would be produced during a save:

ENG	My_Framework_ENG.application My_Framework_ENG.exe.manifest My_Framework_ENG.framework.deploy My_Framework_ENG.app.exe.deploy
FRA	My_Framework_FRA.application My_Framework_FRA.exe.manifest My_Framework_FRA.framework.deploy My_Framework_FRA.app.exe.deploy

So there are four files per language, suffixed with .application, .exe.manifest, .framework.deploy and app.exe.deploy.

What files need to be put onto the HTTP web server?

The four files per language described above and files VF_WB001.HTM and VF_MULTI_YYY.js (where YYY is the language code).

If you are using RAMP VF_SY120.HTM and VF_SY120.JS need to be included.

These files are self contained.

Of course, the WAM functions used by your Framework as filters and command

handlers need to reside on your application server as do any other AJAX HTML files, Style sheets, script files, etc.

Where should these files be served from?

For a developer, from your private folder.

In production environments, your L4Web images folder is the recommended folder.

The server can be an IIS Windows server or a System i Apache HTTP server.

System i Apache server configuration changes

With a System i Apache server some minor configuration changes may need to be made to the MIME types control table so that the files are served correctly:

- Use the Apache administration browser interface (*ADMIN instance in port 2001) and click on Content Settings link.
- In the MIME tab, click on the ADD button to add the following Content-type entries:

File Extension	Value
application	application/x-ms-application
manifest	application/x-ms-manifest
deploy	application/octet-stream

Or if editing the configuration file with a text editor, in the relevant section add these lines:

```
AddType application/x-ms-application application  
AddType application/x-ms-manifest manifest  
AddType application/octet-stream deploy
```

VLF.NET Manual Deployment – An Alternative Way of Deploying VLF.NET

ClickOnce deployment strategy is not viable in some circumstances, such as ones where strict security policy is in place and all applications have to explicitly be given permission to run.

Under ClickOnce deployment method, .NET framework downloads the application files to the current user profile directory and then runs the application from there, making it very hard for the system administrator to define a rule in the security policy to allow the application to run.

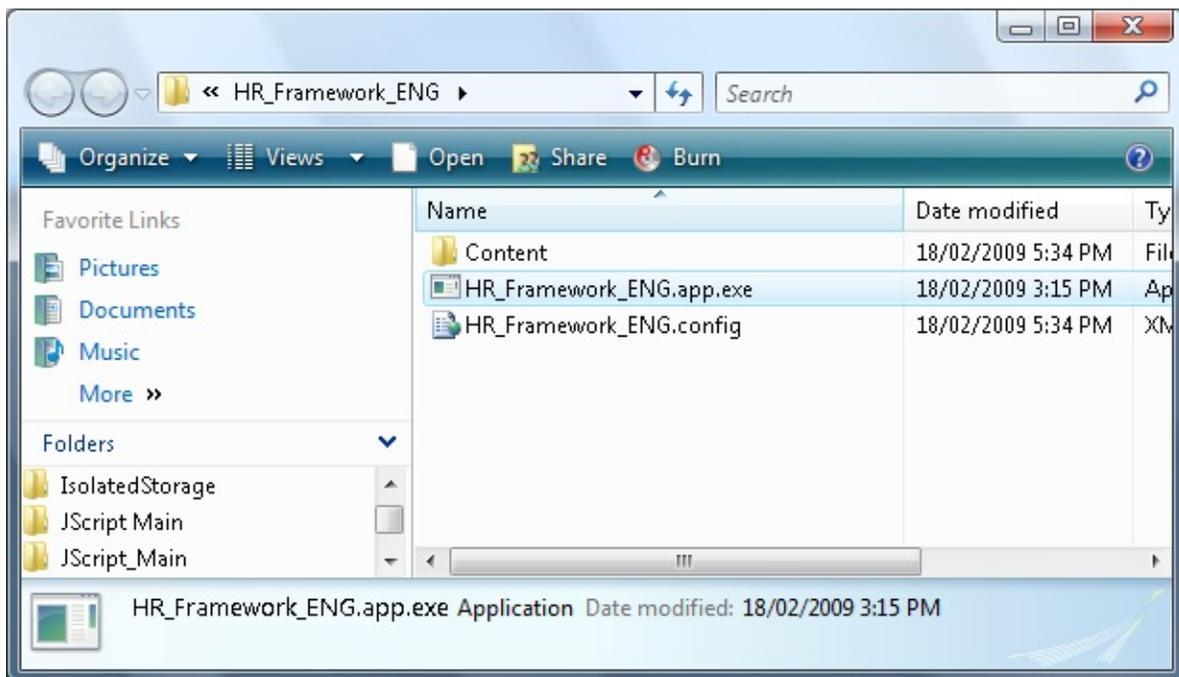
A new feature in this release will address this problem by enabling the network administrator to generate a version of VLF.NET that can be launched locally and does not employ ClickOnce deployment technology. It is important to note however that this alternative deployment method does not change the fact that you will still need to have a LANSA web server to serve the content of your framework.

The generated application files can then be manually copied (XCOPY deployment) to a location accessible by end-users, such as a network share. This way of deploying VLF.NET will also come in handy when Citrix environment is used as it means only one copy of VLF.NET needs to be present in the server as opposed to one copy per user if ClickOnce deployment is used.

Below are the steps to produce VLF.NET application files for manual deployment:

1. Deploy the ClickOnce files as normal to the website hosting VLF.NET
2. Enter in the URL to launch it on the address bar of your browser - make sure to include any additional parameters (such as Partition=xxx) that are required. Add an additional parameter “GenerateDesktopApp=Y”. This parameter is the key to the files generation as instead of running the framework, VLF.NET will generate the application files. It will also write out a configuration file containing all the parameters that you specified in the query string.
3. When completed, VLF.NET will open the folder containing the generated files in Windows Explorer. Those are the files you (or the network administrator) need to manually deploy. Once deployed, the administrator needs to identify the main executable file (it’s the executable file outside the Content subfolder – do not do anything to the files inside the Content subfolder) and do these steps:

- a. Modify the Windows security policy to allow the executable file to run.
- b. Modify the .NET security policy to grant Full Trust to the executable. This is especially important when the files are in a network share as by default they will be treated as partially trusted.
- c. Optional - create a shortcut to the executable file for the convenience of the end-users.



Configuration File

Since the application is going to be launched locally, there is no notion of a URL and query string. To allow you to specify parameters when launching VLF.NET locally, a configuration file is employed as a means to specifying these parameters. The name of the configuration file is the same as the name of the executable file, but ends with .config extension. If you rename the executable file, make sure to rename the configuration file as well to match the executable file.

When you generate the application files (by specifying the parameter `GenerateDesktopApp=Y`) a default configuration file will be generated for you

based on the parameters that you specified in the query string of the URL at the time of generation. If later you need to change the values of the parameters, open the configuration in a text editor (such as notepad) and carefully change the values of the parameters.

Launching a VLF.NET Application

The URL used to launch a VLF.NET application looks just like a normal HTML one. For example:

http://hostname/My_Framework_ENG.application

Also, the VLF URL based parameters like +Trace=Y are passed to VLF.NET applications just like a normal HTML ones are. For example:

http://hostname/My_Framework_ENG.application?Developer=Y+Trace=Y

The URL used to launch a VLF.NET application defaults to partition DEM, so if you are using another partition, you must specify the partition parameter. For example:

http://hostname/My_Framework_ENG.application?partition=EX1

Download size

The minimum download size of a VLF.NET application during the initial download is approximately 6.5 - 8 MB. It will, however, depend on the size of your Framework. Subsequent downloads should be significantly smaller.

How does ClickOnce deployment technology used by VLF.NET work?

When the user enters the application URL in their browser, the browser downloads a small manifest file which it then passes to the .NET runtime. If this is the first time the application is run, .NET runtime will download all the files required to run the application and store it on your local computer. It will then verify the integrity of the application before running it.

The next time the user points their browser to the same URL, .NET runtime will check if the application files have been updated on the server and download the updated or new files as required. It is important to note that your VL Framework version number will be used to determine if there is an updated version of the application available.

It is therefore recommended that the checkbox [Automatically Increment when Saving](#) (Framework -> Properties... -> Identification tab) is ticked so that the version number is incremented automatically with every save.

Allow this Object to be Opened in a New Window

Number of Additional Windows a User can have Open Concurrently:

Multiple Window Control Bar Location:

Your Framework Version Number . . .

Automatically Increment when Saving

Show in Help About Text

Last Changed

If this option is not selected, you will need to change the version number yourself whenever you are uploading the files to the webserver.

Digital Certificates

How are .NET applications signed at compile time?

If you have a digital certificate that is issued by a certification authority (such as VeriSign) you can sign your VLF.NET application to ensure their authenticity to end-users. To do this you need to specify the [Certificate File \(PFX\)](#) and optional [Certificate File Password](#) in the Developer Preferences tab. These are used when you save your Framework to digitally sign your VLF.NET application.



The image shows a screenshot of the ".NET Compilation Options" dialog box. It has a light blue border and a white background. At the top left, the title ".NET Compilation Options" is displayed. Below the title, there are two text input fields. The first field is labeled "Certificate File (PFX)" and the second is labeled "Certificate File Password". To the right of these fields is a button labeled "Verify".

The certification authority supplied my certificate in two files: .PVK and .SPC (or .PVK and .CER). Can I use them?

Yes but you will need to create a Personal Information Exchange (PFX) file first from those 2 files. Use pvk2pfx tool to do so. The information about this tool can be found on <http://msdn.microsoft.com/en-us/library/aa906332.aspx>.

What happens if a valid digital certificate is not used?

If a valid digital certificate (a digital certificate issued by a certification authority) is not used, the application will be signed with an auto-generated temporary digital certificate. The application can still be deployed and run, however the first time the application is run by a user, a dialog box similar to below will appear warning the user that the publisher of the application can't be verified.

Please note that the temporary digital certificate is generated for testing purposes only and VLF.NET applications signed with this temporary certificate should not be used in production environment.



Is VLF.NET a Full Trust or Partial Trust .NET application?

VLF.NET is a Full Trust .NET application. This means that VLF.NET is not subjected to code access security checking.

The ClickOnce deployment process will take care of this automatically and it is not necessary for the users or administrators to configure their computers.

If a valid certificate issued by a certification authority is used, the first time the application is run by a user a dialog box will appear, warning the user that the application requires access to the computer that is potentially unsafe.



I'm using a valid certificate issued by a certification authority. Is there a

way to prevent the user from being prompted with the potentially unsafe access security warning dialog box?

Yes, your certificate will need to be added to the 'Trusted Publisher' section of every computer that is going to run VLF.NET. Your domain administrator should be able to configure this easily for you.

How are .NET applications verified at execution time?

VLF.NET applications are downloaded using a Microsoft .NET web based deployment technology called 'ClickOnce'. The download procedure it uses automatically verifies the digital certificate associated with your application.

Can I run VLF.NET over SSL (HTTPS)?

Yes, but you will need an SSL certificate issued by a trusted Certification Authority (such as VeriSign) installed on your web server. Below is an extract from MSDN in regards to ClickOnce deployment over SSL:

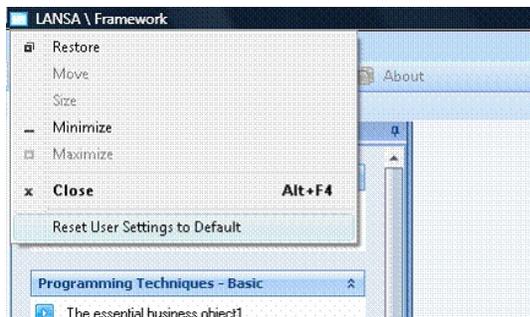
A ClickOnce application will work fine over SSL, except when Internet Explorer raises a prompt about the SSL certificate. The prompt can be raised when there is something wrong with the certificate, such as when the site names do not match or the certificate has expired. To make ClickOnce work over an SSL connection, make sure that the certificate is up-to-date, and that the certificate data matches the site data.

VLF.NET Form Layout Details

VLF.NET form layout details are remembered between sessions.

They are stored in a file under a directory called VLF, which is a subdirectory under the roaming profile application data folder of the current Windows user. The file is named after the Framework name with .xml extension. The layout details can be reset by deleting this file.

You can also reset the layout details easily from the application window menu. To open the window menu, click on the application icon. Click on 'Reset User Settings to Default'. Note that it will also clear the most recently used business object list if it is enabled for the Framework.



If it is installed on C: drive and your Framework name is MyFramework, the settings file is called MyFramework.xml and can be found under C:\Users\UserName\AppData\Roaming\VLF.

How are the form layout details stored?

They are stored by user, not device, and they are stored for individual Frameworks.

Visual Themes

VLF.NET supports five themes:

- Current Windows theme
- 2007 Blue
- 2007 Silver
- 2007 Graphite
- 2007 Olive

If one of the 2003 themes is selected, it will be replaced with the equivalent 2007 theme (e.g. 2003 Silver will be mapped to 2007 Silver).

See [Web Application Start Options](#).

Windows Framework Properties Applicable to VLF.NET

Some Windows Framework and business object properties which are ignored by VLF.WEB are observed by VLF.NET.

Framework - Details Tab

Properties	Behavior in VLF.NET
Allow Panes to be Shrunk and Expanded	Allow panes to be pinned and unpinned
Enable the Position Menu Option	Allow panes to be docked, undocked, and dragged around
Allow Search/Recently Used Limit	Same as in Windows Framework applications
Search Field Width	Same as in Windows Framework applications

Command Display Tab

Properties	Behavior in VLF.NET
Command Tab Style	Supports Tabs, Buttons, and FlatButtons (Stacked is not supported)
Command Tab Location	Same as in Windows Framework applications

Business Object – Instance List / Relations Tab

Properties	Behavior in VLF.NET
Instance List Tool Bar Location	Same as in Windows Framework applications

Business Object – Filter Settings Tab

Properties	Behavior in VLF.NET
Filter Tab Style	Same as in Windows Framework applications
Filter Tab Location	Same as in Windows Framework applications

Framework-AJAX Applications

To understand what AJAX applications are, see <http://en.wikipedia.org/wiki/AJAX> for a generic description.

Benefits of AJAX applications

See <http://en.wikipedia.org/wiki/AJAX> for a general description of the benefits of an AJAX application.

In the Framework, AJAX applications offer:

- Optimal web performance by avoiding full page or full frame refreshes
- The ability to create web forms that are close to Windows rich client in function and speed.
- On-the-fly data validation
- On-the-fly data fetching/updating/deleting
- Maximum control and freedom in processing logic and UI interactions

Disadvantages of AJAX applications

Compared to WAM applications, AJAX application development requires an extended skill set and usually longer development times.

Also, AJAX applications generally produce higher web and application server hit rates. The requests are usually simpler, but made more often. This changes the profile of the performance load on your web and application servers. This is a very important consideration in your application design.

When should I use an AJAX application?

Generally, you should only use AJAX applications when the additional cost of developing them is less than the value of the benefit they deliver (the problem of course is in assessing the true value of the benefit they deliver).

A typical Framework-WEB application should mainly consist of WAM functions, with the occasional AJAX routine used in selected application areas.

You might use AJAX:

- When optimal performance is required (for example high volume data entry)

- When optimal or elaborate UI interactions are required (for example order entry)
- As demonstration vehicles for customer or end-users
- In areas that are very heavily used by customer or end-users

[Mandatory Skills](#)

[AJAX Applications in the Framework](#)

[AJAX Exercises](#)

[Tracing](#)

[Messages and Errors](#)

[Frequently Asked Questions](#)

[Restrictions](#)

[Recommendations](#)

Mandatory Skills

To create AJAX applications you must have:

- At least basic JavaScript and HTML or DHTML programming skills
- LANSAR DHTML or DHTMLX programming skills

The material in this guide assumes that you have these skills already.

You do not need any XML or XSLT skills.

Getting Started

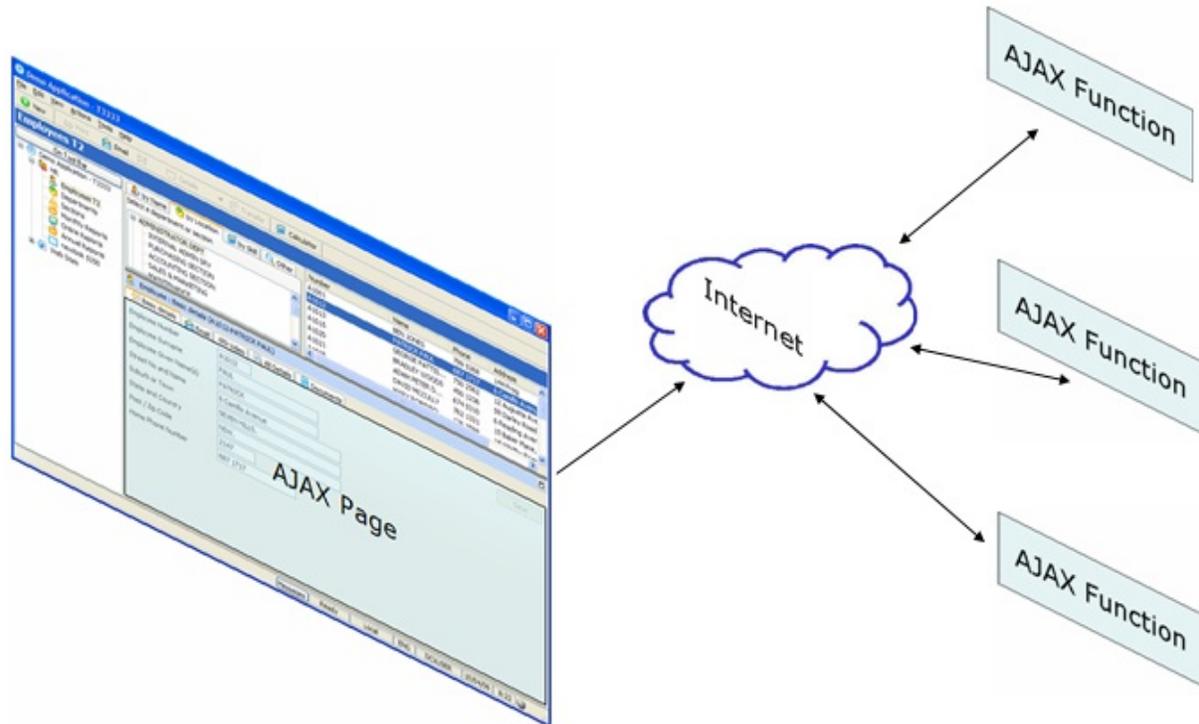
If you do not have basic JavaScript or HTML/DHTML skills you should build them up first using simple examples. There are many excellent books and web sites that can help you acquire these basic skills. For example

<http://www.w3schools.com/>.

If you already have these skills and wish to apply them in the Framework-AJAX context, the best place to start is with the [AJAX Exercises](#) in this guide.

AJAX Applications in the Framework

You can create filters and command handlers as AJAX routines (just as you use VL components or WAMs to create filters and command handlers).



An AJAX routine usually consists of at least two parts:

An AJAX page This is an HTML/JavaScript document that visualizes the filter or command handler inside the Framework. It manages all the interactions between the user, the Framework and the AJAX functions executing on the server(s).

One or more AJAX functions These are normal RDML or RDMLX functions that execute on the application server. They use AJAX processing to communicate with the AJAX page. An AJAX page will often use multiple AJAX functions. Sometimes AJAX functions service multiple AJAX pages.

They are not WAMs or VL components.

AJAX Pages and Functions

Interaction between AJAX pages and AJAX functions

Information Flow Between AJAX Pages and Functions

Things an AJAX pages must provide to the Framework

Things the Framework Provides to an AJAX page

AJAX Pages and Functions

AJAX Pages

An AJAX page is an HTML document. Typically it contains JavaScript to manage interactions between the application server and the user interface.

For example, an HTML page may display a “Search” button on the user interface.

When the user clicks it, JavaScript code (executing on the client PC) assembles a request and sends it to the application server.

The AJAX function (executing on the server) receives the request, processes it, and sends back a response.

The JavaScript code in the AJAX page receives the response and updates the user interface, for example by dynamically displaying a list of customers.

There are two significant things about this interaction:

- The request sent to the application server is handled asynchronously. This allows the user and JavaScript to do other things while waiting for the server to respond.
- The JavaScript typically dynamically rebuilds just part of the user interface (for example a list) rather than rebuilding the entire HTML page or frame from scratch.

AJAX Functions

AJAX functions are normal RDML or RDMLX functions, so they are architecturally different to WAMs. They have no user interface capabilities. Their only purpose is to receive information from and return information to AJAX page(s).

The information is received and returned as pure data.

The AJAX page manages how the information is input by, or displayed to, the user.

Where do AJAX pages and functions reside?

AJAX pages reside on your web server, typically in your Framework private folder while you are doing development.

AJAX functions reside on your application server.

Typically your web server and application server are on the same computer.

Where do AJAX pages and functions execute?

The JavaScript in AJAX pages execute inside a web browser on a client PC.

The RDML or RDMLX code inside an AJAX function executes on your application server.

Very rarely do your web browser and application server reside on the same computer. The most notable exception is when you are developing applications using the VL-IDE on your PC and executing them under the IIS web server which is also executing on your PC.

Interaction between AJAX pages and AJAX functions

The Framework model for exchanging information between the JavaScript in an AJAX page and the RDML(X) code in an AJAX module simply re-uses the Framework's Virtual Clipboard model.

Here's a simple example of an AJAX interaction:

Product Description

An AJAX page has presented a form with a Product Description and a Search button. When the user clicks the button, a list of products that contain the description is displayed.

For example the user enters "BOLTS" as the product description and clicks Search.

The **AJAX Page** JavaScript, executing on the client, handles the Search button click and:

- Saves the value "BOLTS" with the name "Description" onto the virtual clipboard.
- Sends a request to the application server. The request contains:

The name of the AJAX function to be invoked on the server

The action that the AJAX function should take (for example PRODUCTSEARCH)

The JavaScript function to call when the request has been completed

It then does nothing more. The search request response will come back from the server asynchronously when it has been completed.

The **AJAX function** is invoked on the application server:

- It gets the request that the client made (PRODUCTSEARCH).
- It gets the value of the name "Description" from the clipboard ("BOLTS").
- It finds all the products containing the word "BOLTS" and puts their Numbers and Descriptions and Quantity on Hand values onto the clipboard in a list.

- It completes execution.

The **Framework** detects the completion of the server AJAX function. It then calls the JavaScript function (in the AJAX page) that was nominated to handle the response. The JavaScript function then:

- Removes all the <TR> table rows for a product table is it is displaying on the web page.
- Reads from the clipboard the list of product numbers, descriptions and quantity on hand values that the AJAX function put there. From this information it formats a new set of <TR> table rows and displays them to the user.

Product Description

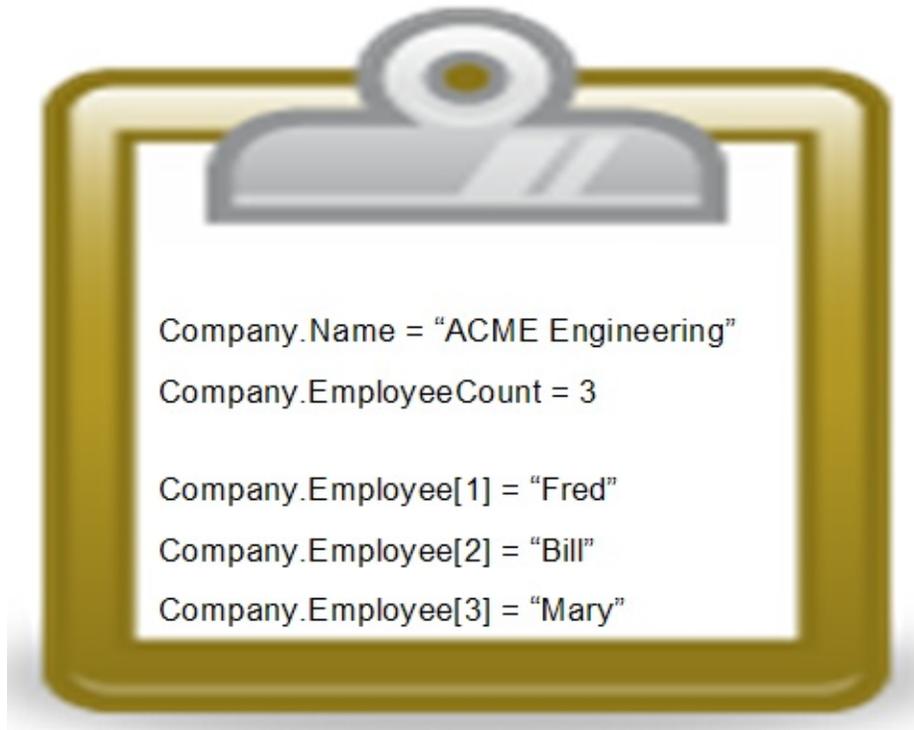
Number	Description	QOH
0012345	1/4 inch Hex Bolt	16 box - 32 per box
0026273	24mm Hex Bolt	9 box - 144per box
0043647	1/4 inch S/S Bolt	6 box - 32 per box
0016273	1/4 inch Slotted Bolt/Nut	160 blister pack - 1 per pack
0075858	1/4 inch Zinc Plated Bolt	8 box - 12 per box

Almost all AJAX Page - AJAX function interactions follow this basic flow of control model. What they actually do of course varies widely.

The magic of AJAX is that this small Product Search area may be imbedded inside a much larger and more complex AJAX page. When the search is executed, only the Product Search area is refreshed by the browser, not the entire page.

Information Flow Between AJAX Pages and Functions

The primary means of sharing information between AJAX pages, AJAX functions and RAMP scripts is the virtual clipboard.



The virtual clipboard is essentially a namespace of names, values and lists:

- JavaScript code (executing on the client PC) in your AJAX pages can read or write information from/to the virtual clipboard.
- RDML/RDMLX code (executing on the web server) in your AJAX functions can read or write information from/to the virtual clipboard.
- RAMP JavaScript code (executing on the client PC) in your AJAX pages can read or write information from/to the virtual clipboard.

The clipboard they all use is shared. It is created when the Framework-WEB session is started and persists until the session ends, when it is destroyed. Clipboard content changes are invisibly transferred between the client PC and web server (or vice-versa) as required.

Only the clipboard changes are transferred, but of course the number of changes you make impacts how much information needs to be transferred.

Some good practice tips for using the virtual clipboard are:

- You should develop and document standards for the names and values it contains.
- The amount of clipboard data you transfer from the client to the server particularly impacts application performance. Transferring large amounts of data from the server to the client has less impact.

AJAX Information in the Framework Virtual Clipboard

When you are using AJAX applications, the Framework puts some information onto the virtual clipboard automatically. Generally you should treat this information as read-only so as to avoid interfering with the operation of the Framework.

Name Part 1	Name Part 2	Name Part 3	Usage Comment
AJAX	SYSTEM	AJAXPAGE	Current AJAX page name
AJAX	SYSTEM	AJAXMODULE	Default RDML or RDML AJAX function name
AJAX	SYSTEM	REQUEST	Current request
AJAX	SYSTEM	PAYLOAD	Current payload
AJAX	SYSTEM	RETURNCODE	Latest Return Code
AJAX	SYSTEM	MESSAGECOUNT	Latest Message Count (can be updated)
AJAX	SYSTEM	MESSAGE	Latest Message Instance
AJAX	SYSTEM	<any other>	Do not use this namespace. It is reserved for the current and future Framework versions.

Typically you will only ever need to access `AJAX.SYSTEM.REQUEST` and `AJAX.SYSTEM.PAYLOAD` in your server side AJAX functions like this:

* Get the action and payload that the Javascript sent ...

```
Execute Subroutine(GETA) With_Parms(SYSTEM REQUEST 1 #REQUEST)  
Execute Subroutine(GETA) With_Parms(SYSTEM PAYLOAD 1 #PAYLOAD
```

* Now switch on the requested action

```
Case Of_Field(#REQUEST)
```

* Load 10 sample entries

```
When Value_Is(= LOAD10)  
Execute Subroutine(Load10)
```

* Validate a zip code

```
When Value_Is(= VALIDATE)  
Execute Subroutine(VALIDATE)
```

* Handle a bad request

```
Otherwise
```

```
Abort Msgid(DCM9899) Msgf(DC@M01) Msgdta(('Unknown request ' + #RE
```

```
Endcase
```

Typically this allows one AJAX function to handle many different AJAX page requests coming from many different AJAX pages.

Things an AJAX pages must provide to the Framework

When you create an HTML document that is to be used by the Framework as an AJAX page, it needs to expose three JavaScript functions to the Framework like this:

```
<HTML>
<HEAD>
<link rel='stylesheet' type='text/css' href='vf_vs001.css' >
<script>
/* =====
/* ===== Handle Page Initialization =====
/* =====
function VF_AJAX_Initialize()
{
    return;
}
/* =====
/* ===== Handle Page Execution =====
/* =====
function VF_AJAX_Execute()
{
    return;
}
/* =====
/* ===== Handle Page Termination =====
/* =====
function VF_AJAX_Terminate()
{
    return;
}
...etc...
...etc...
```

Once this has been done the AJAX page can be used by the Framework. See [Exercise 1: AJAXEX1 – Hello World](#) for an example of doing this.

Things the Framework Provides to an AJAX page

The Framework formally exposes various JavaScript objects for use by the AJAX page:

Name	Type	Comments
Using BUSY and SETBUSY	Function	Sets the Framework busy state to true or false
Using BUSY and SETBUSY	Function	Queries the current Framework state. Returns true or false
SENDREQUEST Function	Function	Sends a request to an AJAX function on the server
AVSAVEVALUE Function	Function	Saves a named value to the virtual clipboard
AVRESTOREAVALUE and AVRESTORENVALUE Functions	Function	Restores an alphanumeric named value from the virtual clipboard
AVRESTOREAVALUE and AVRESTORENVALUE Functions	Function	Restores a numeric named value from the virtual clipboard
Using STYLESHEET	Function	Returns the name of the variable style sheet that should be used for the user selected WINDOWS/XP/WEB look. XP WEB look is no longer available.
Using AJAXGLOBAL to Share Information between Pages	Object	Global object for exchanging information between AJAX pages.
You can add tracing to your AJAX pages by using the supplied AVTRACE function.	Function	Add information to the trace record

--	--	--

Using BUSY and SETBUSY

BUSY

BUSY() returns True or False indicating whether the Framework is currently in a busy state.

Usually, but not always, you want to ignore a user’s request if the Framework is busy.

SETBUSY

SETBUSY(True or False) allows you to set the Framework’s busy state.

Usually, but not always, you want to use SETBUSY(True) just before sending a request to the server and then use SETBUSY(False) when the server operation completes.

Using SENDREQUEST

SENDREQUEST Function

Sends an asynchronous request to an AJAX function on the web server.

Syntax

SETBUSY(SenderWindow,Function,Request,Payload,ResponseHandler,UserOb

Parameters

SenderWindow	Required. Always pass as window so as to identify the window from which the request originated.
Function	Required. The name of the AJAX style RDML or RDMLX function to be invoked on the server. If passed as null or "" it will default to the name of the function defined in the Framework along with this page.

Request	Required. The request that is to be passed into the AJAX function. Typically this value is used so that a single AJAX function can handle multiple different requests.
Payload	Required. The payload that is to be passed into the AJAX function. Typically used to qualify the Request with additional information. If passed as null or "" it will be passed into the AJAX function as blanks. Maximum length 256.
ResponseHandler	Required. The JavaScript function that is to receive control when the AJAX function completes execution.
UserObject	Optional. Any JavaScript object that should be passed into the Handler. Typically used to qualify to the handler function the visual item it should update.
ReportErrors	Optional. Indicates whether fatal errors detected in the AJAX function should be reported by the Framework. Passed as True or False. Default is True.
RouteMessages	Optional. Indicates whether messages returned by the AJAX function should be automatically routed into the Framework message area. Pass as True or False. Default is True.

Return Value

None.

Handle SENDREQUEST Responses

When you issue a SENDREQUEST operation you must nominate the JavaScript response handler function that is to receive control when the AJAX function completes execution:

SETBUSY(SenderWindow,Function,Request,Payload,ResponseHandler,UserOb

The response handler JavaScript function must be declared like this:
function

MyHandler(Function,Request,Payload,UserObject,FatalError,FatalMessage)

These parameters are always passed into the response handler by the Framework:

Function	The name of the AJAX function that has completed execution.
Request	The request that the AJAX function handled.
Payload	The payload that was presented to the AJAX function.
UserObject	The UserObject that was passed into the SENDREQUEST function that initiated this response.
FatalError	Contains True or False indicating whether a fatal error was detected when executing the AJAX function.
FatalMessage	A composite message containing all the error messages returned by an AJAX function that experienced a fatal error. If no fatal error was detected this value is passed as "".

Using AVSAVEVALUE and AVRESTORExVALUE

These are like other virtual clipboard access routines used elsewhere in the Framework.

AVSAVEVALUE Function

Saves an alphanumeric or numeric value onto the Framework virtual clipboard.

Syntax

AVSAVEVALUE(vValue, sID1, sID2, sID3, iInstance, sLanguage)

Parameters

vValue	Required. Alphanumeric or numeric value to save to the virtual clipboard.
--------	---

	<p>If this parameter is a JavaScript variable of type string, the value is posted to the clipboard as an alphanumeric value and can therefore only be sensibly retrieved using the AVRESTOREAVALUE function (or equivalent).</p> <p>If it is of type number, it is posted as type numeric to the clipboard and can only be sensibly retrieved using the AVRESTORENVALUE function (or equivalent).</p>
sID1	Required. String that contains the Virtual Clipboard identifier 1.
sID2	Optional. String that contains the Virtual Clipboard identifier 2.
sID3	Optional. String that contains the Virtual Clipboard identifier 3.
iInstance	Optional. Integer that contains the instance number. Defaults to 1 when not specified. Instances are typically used to create lists of clipboard values and usually accompanied by another clipboard value that indicates how many entries currently exist in the list.
sLanguage	Optional. String that contains the language code. Defaults to ALL languages when not specified.

Return Value

None.

AVRESTOREAVALUE and AVRESTORENVALUE Functions

Restore an alphanumeric or numeric value from the Framework virtual clipboard.

Syntax

AVRESTOREAVALUE/AVRESTORENVALUE(Default, sID1, sID2, sID3, iInstance, sLanguage)

Parameters

Default	Required. String/Number that contains the default value to return
---------	---

	if the value is not found.
sID1	Required. String that contains the Virtual Clipboard identifier 1.
sID2	Optional. String that contains the Virtual Clipboard identifier 2.
sID3	Optional. String that contains the Virtual Clipboard identifier 3.
iInstance	Optional. Integer that contains the instance number. Defaults to 1 when not specified
sLanguage	Optional. String that contains the language code. Defaults to ALL languages when not specified.

Return Value

None.

Using STYLESHEET

STYLESHEET returns a script which contains the name of the main cascading style sheet associated with the current Framework web session.

It has no arguments.

Using AJAXGLOBAL to Share Information between Pages

AJAXGLOBAL is a JavaScript object that is only accessible to AJAX pages. You can use it to share information between AJAX pages.

For example, if AJAXPAGE01.HTM did this:

```
AJAXGLOBAL.CurrentCompany = "ACME Engineering"
```

then AJAXPAGE002.HTM could do this:

```
alert(AJAXGLOBAL.CurrentCompany);
```

Use code like:

```
if (typeof(AJAXGLOBAL.XXXXXX) == "undefined")
```

to check whether object XXXXXX exists in AJAXGLOBAL.

AJAXGLOBAL is a very efficient way to share information between AJAX pages, but it is only accessible to AJAX page scripts, so if you want share information with server AJAX functions or RAMP scripts you should use the virtual clipboard instead.

AJAXGLOBAL persists from session start to session end at which time it is destroyed.

AJAX Exercises

Exercise 1: [AJAXEX1 – Hello World](#)

Exercise 2: [AJAXEX2 – Doing Some Simple AJAX](#)

Exercise 3: [AJAXEX3 – Using Asynchronous Processing](#)

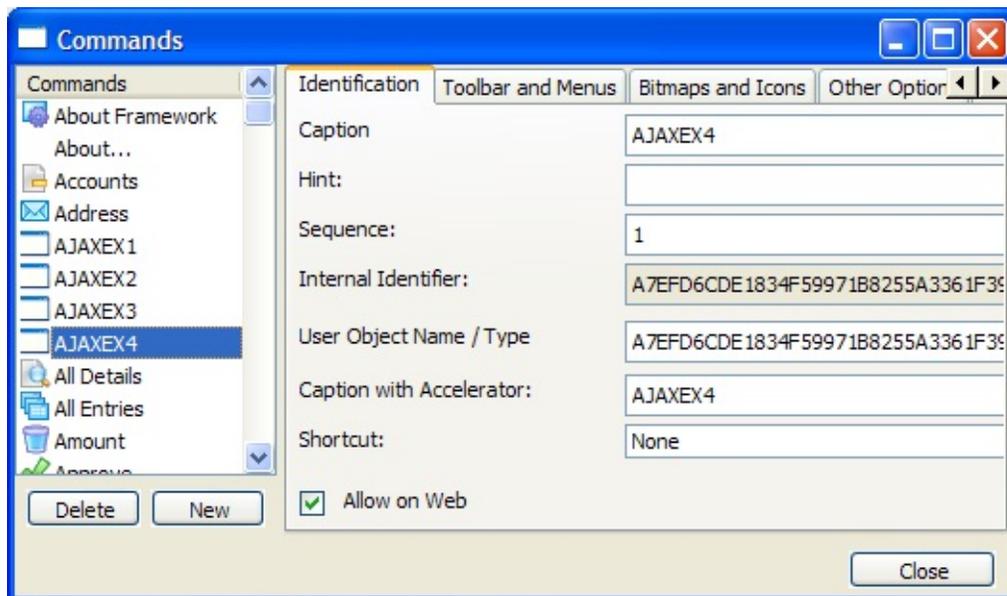
Exercise 4: [AJAXEX4 – Using the Instance List](#)

Exercise 1: AJAXEX1 – Hello World

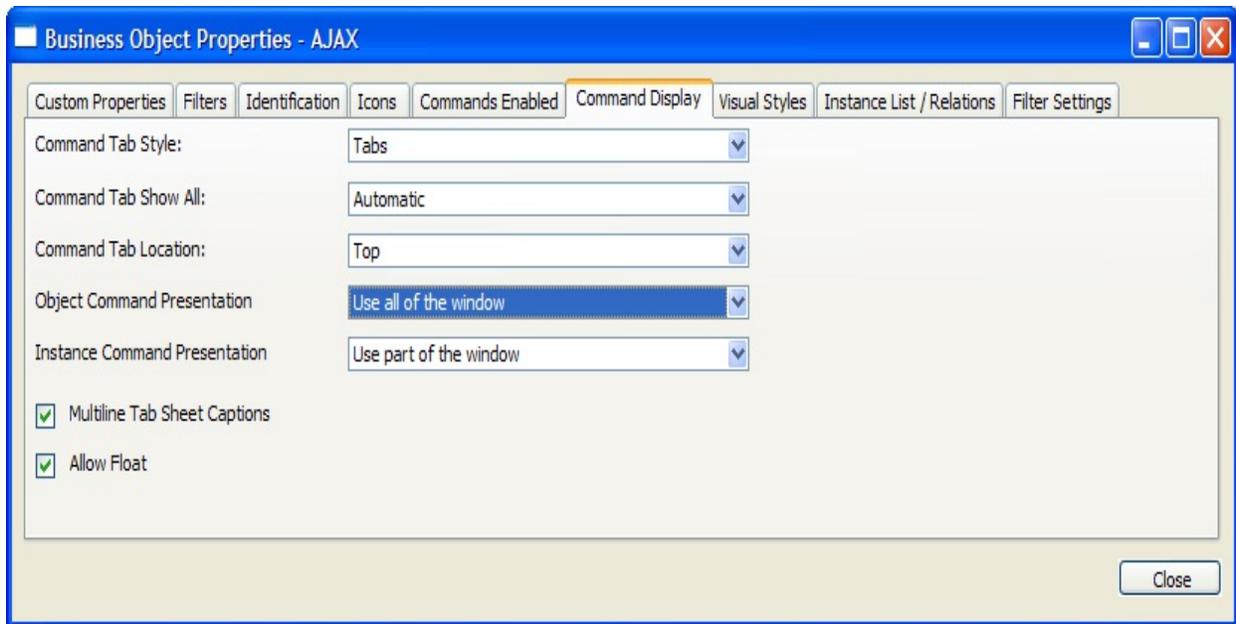
[AJAX Page AJAXEX1.HTM](#)

The Steps

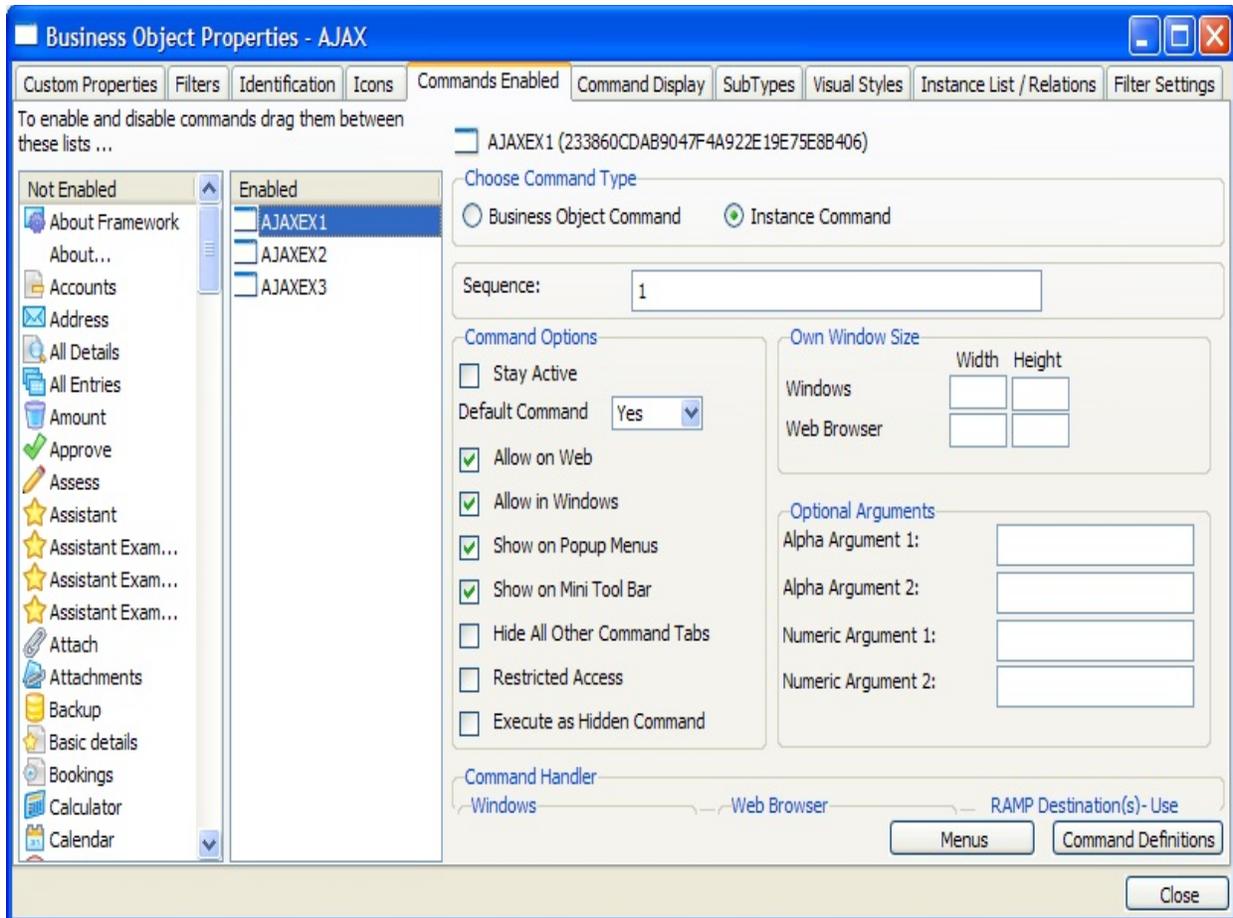
- In your Framework create four new commands named AJAXEX1, AJAXEX2, AJAXEX3 and AJAXEX4. Use (Framework) -> (Commands) to do this:



- Manually create an application named 'AJAX Examples'
- In application 'AJAX Examples', manually create a business object named 'AJAX'.
- Delete all filters from object AJAX. It is not going to have any filters or an instance list.
- Change business object AJAX so that its Object Command Presentation option is Use all of Window:

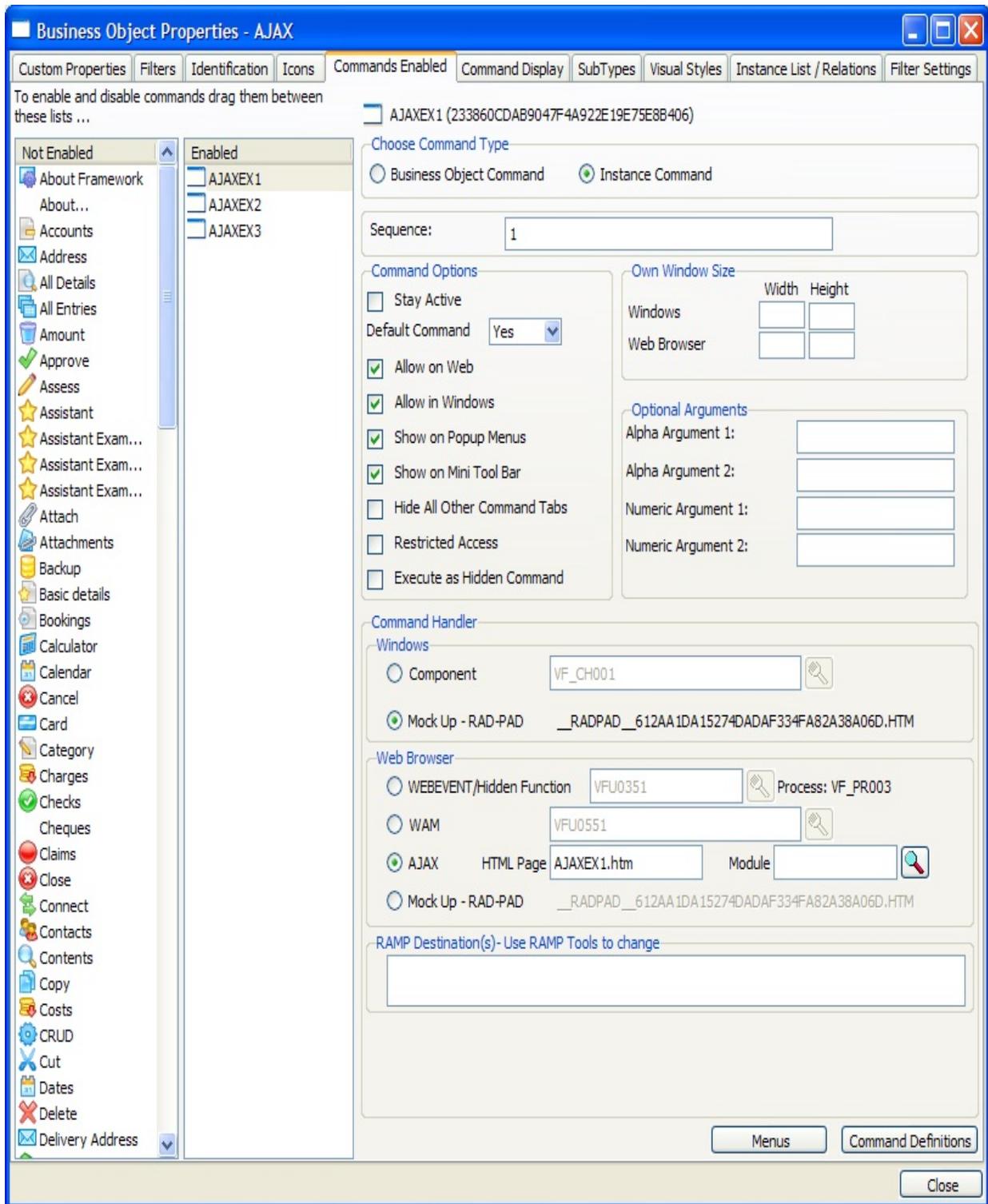


- In the AJAX business object enable the commands AJAXEX1, AJAXEX2 and AJAXEX3. These should all be business object level commands, because object AJAX is not going to have an instance list associated with it.
- Make command AJAXEX1 the default command for the object AJAX. Save your Framework and restart it:

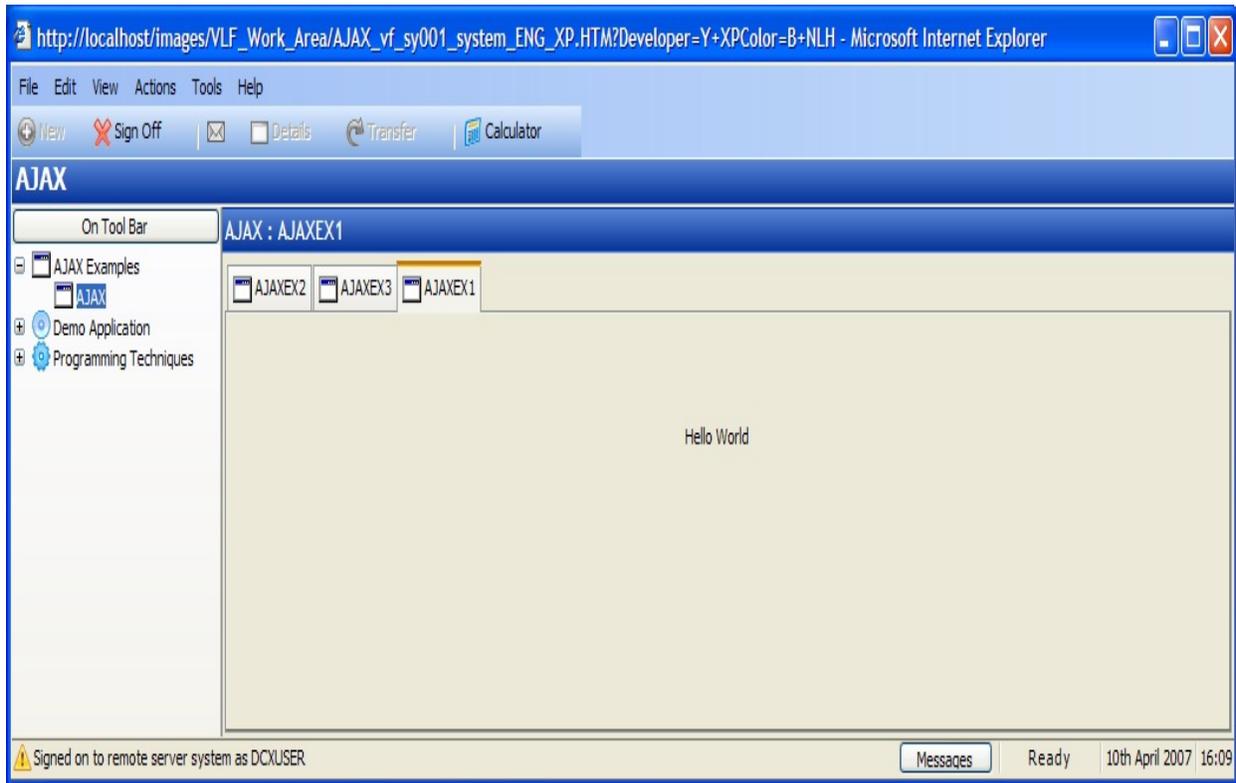


Copy and paste the DHTML-JavaScript code in [AJAX Page AJAXEX1.HTM](#) using NOTEPAD and save it into your Framework private working folder as an AJAX page file named AJAXEX1.HTM.

Now snap AJAX page file AJAXEX1.HTM into your Framework as the command handler to be associated with command AJAXEX1 like this:



Save, restart and execute your Framework in a web browser. Select application 'AJAX Examples' and then select business object AJAX. This should execute the default command AJAXEX1, causing a 'Hello World' display like this:



AJAX Page AJAXEX1.HTM

```
<HTML>
<HEAD id='HEAD_Tag'>
<link rel='stylesheet' type='text/css' href='vf_vs001.css' >
<script>
/* =====
/* ===== Handle Page Initialization =====
/* =====
function VF_AJAX_Initialize()
{
  /* Insert the variable style sheet (ie: XP, WIN, WEB style) into this page */
  {
    var objLink = document.createElement("LINK");
    objLink.rel = "stylesheet"; objLink.type = "text/css";
    objLink.href = STYLE_SHEET();
```

```
    document.getElementById("HEAD_Tag").insertAdjacentElement("afterBe
}
/* Finished */

return;
}
/* =====
/* ===== Handle Page Execution =====
/* =====
function VF_AJAX_Execute()
{
    SETBUSY(false);
    return;
}
/* =====
/* ===== Handle Page Termination =====
/* =====
function VF_AJAX_Terminate()
{
    return;
}
/* =====
</script>
</HEAD>
<BODY id='HEAD_Tag'>
<br/><br/><br/>
<P align='center'>Hello World</p>
</BODY>
</HTML>
```

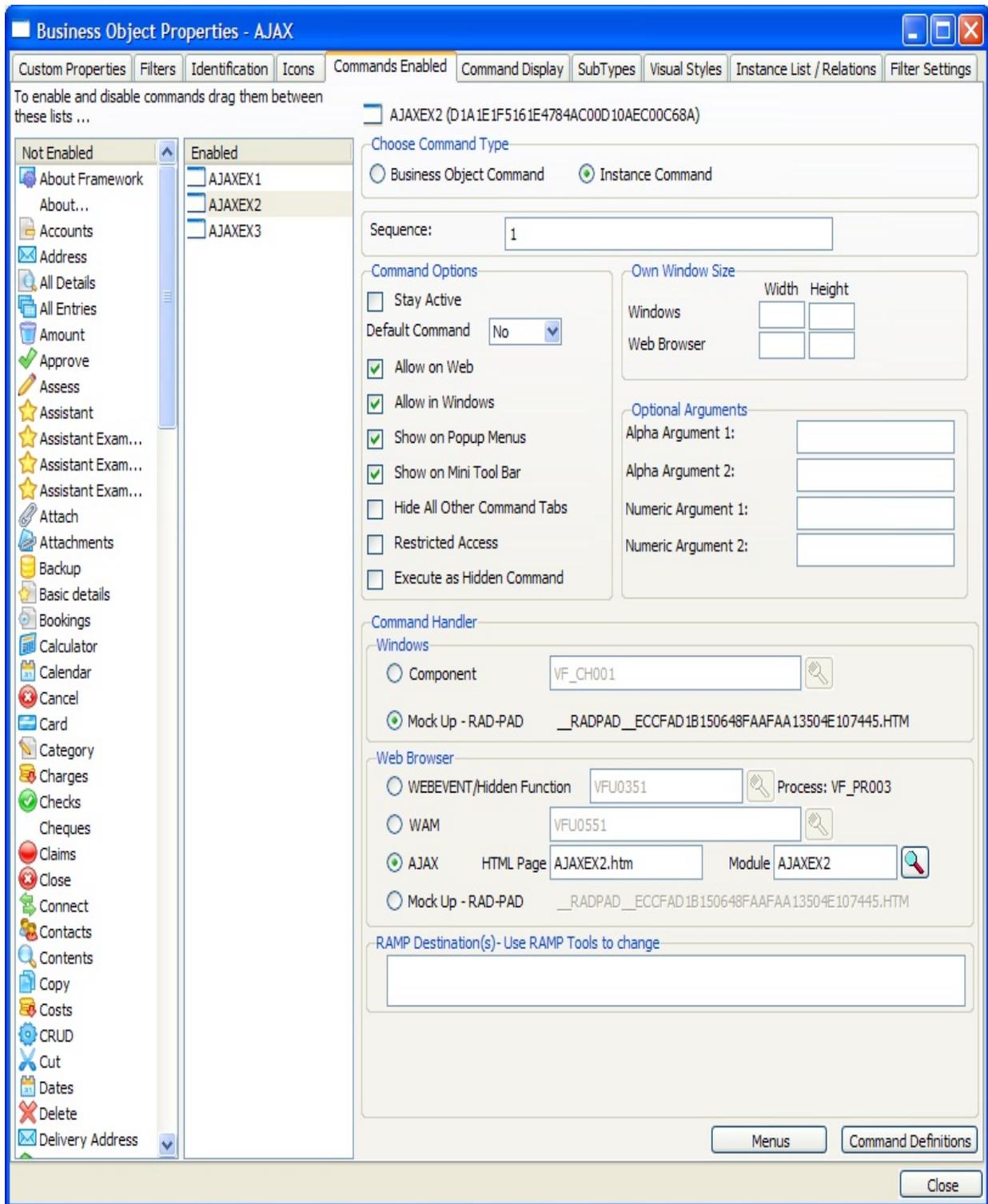
Exercise 2: AJAXEX2 – Doing Some Simple AJAX

[AJAX Page AJAXEX2.HTM](#)

[AJAX Function AJAXEX2](#)

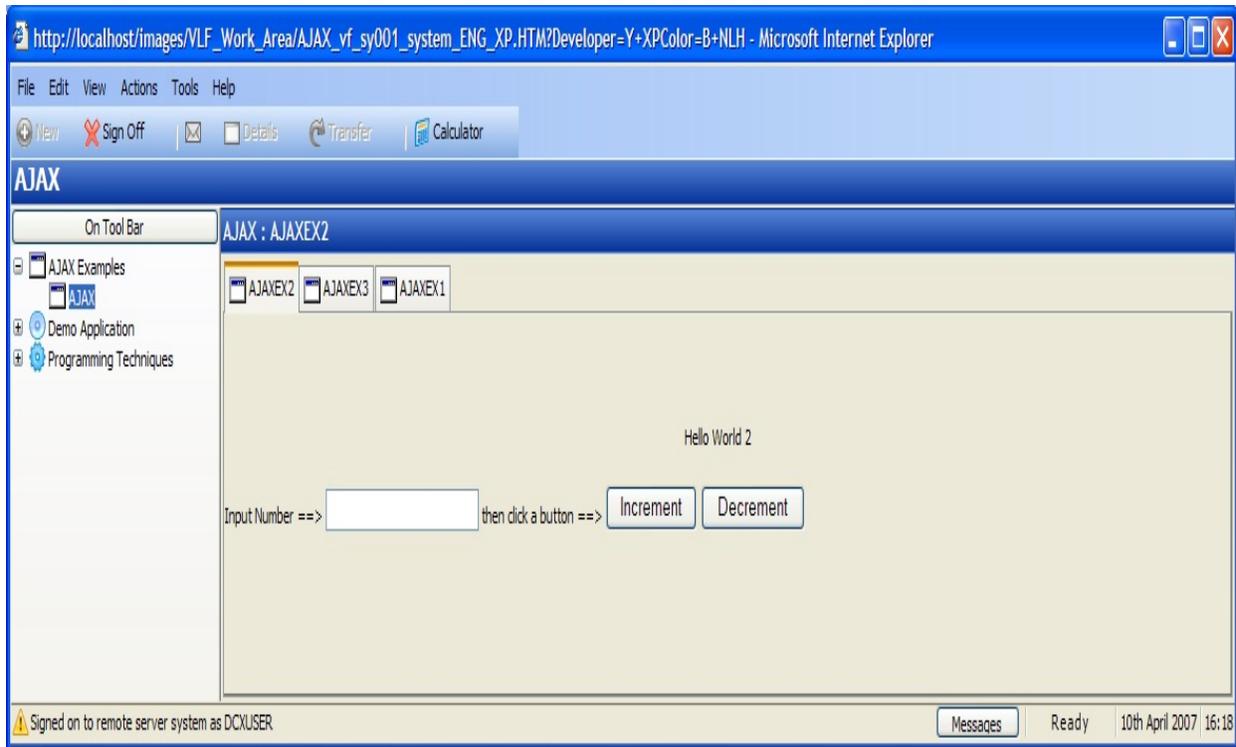
The Steps

- Copy and paste the DHTML-JavaScript code in [AJAX Page AJAXEX2.HTM](#) using NOTEPAD and save it into your Framework private working folder as an AJAX page file named AJAXEX2.HTM.
- Copy and paste the AJAX function code in [AJAX Function AJAXEX2](#) into an RDML function named AJAXEX2 into your VL-IDE. Compile RDML function AJAXEX2 so that it is executable on your web server.
- In business object AJAX, snap AJAX page file AJAXEX2.HTM and AJAX function AJAXEX2 into your Framework as the command handler to be associated with command AJAXEX2 like this:



Save, restart and execute your Framework in a web browser. Select application 'AJAX Examples' and then select business object AJAX. This should execute

the default command AJAXEX1. Click on the AJAXEX2 tab to execute the AJAXEX2 command:



Enter a number and then click the increment and decrement buttons. The value you enter is sent to the AJAX function on the web server and incremented or decremented as requested. The result is then redisplayed in the AJAX page.

Sending a number to a web server to be incremented or decremented is only to illustrate the foundations upon which all AJAX applications are built, including:

- Sending request to the server
- Receiving responses from the server
- Asynchronous processing
- Dynamic updating of the web page

You should review the JavaScript/DHTML code in AJAXEX2.HTM and the RDML code in AJAX function AJAXEX2 until you understand the basics of what it is doing and how it is doing it.

AJAX Page AJAXEX2.HTM

```
<HTML>
<HEAD id='HEAD_Tag'>
<link rel='stylesheet' type='text/css' href='vf_vs001.css' >
<script>
/* =====
/* ===== Handle Page Initialization =====
/* =====
function VF_AJAX_Initialize()
{
  /* Insert the variable style sheet (ie: XP, WIN, WEB style) into this page */
  {
    var objLink = document.createElement("LINK");
    objLink.rel = "stylesheet"; objLink.type = "text/css";
objLink.href = STYLESHEET();
    document.getElementById("HEAD_Tag").insertAdjacentElement("afterBe
  }

  /* Finished */
  return;
}
/* =====
/* ===== Handle Page Execution =====
/* =====
function VF_AJAX_Execute()
{
  SETBUSY(false);
  return;
}
/* =====
/* ===== Handle Page Termination =====
/* =====
function VF_AJAX_Terminate()
{
  return;
}
```

```

/* ----- */
/* Rationalized virtual clipboard access to name space AJAX */
/* ----- */
function Put(val,np2,np3,inst) {AVSAVEVALUE(val,"AJAX",np2,np3,inst);}
function GetN(np2,np3,inst) {return(AVRESTORENVALUE(0,"AJAX",np2
/* ----- */
/* Handle a click of button BUTTON_onclick */
/* ----- */
function BUTTON_onclick(strRequest)
{
/* Ignore if already busy doing something else */
if (BUSY()) return;

/* Validate the number and put it onto the clipboard */
{
var floatNumber = parseFloat(NUMBER_Tag.value);
if (isNaN(floatNumber)) { alert("Invalid number entered"); return; }
Put(floatNumber,"NUMBER");
}

/* Make busy now */
SETBUSY(true);

/* Update the message area */

MESSAGE_Tag.innerText = "Processing your request now. Please wait.";
MESSAGE_Tag.style.backgroundColor = "orange";

/* Now send INCREMENT/DECREMENT request the server RDML functio

if (strRequest == "I" ) SENDREQUEST(window,"","INCREMENT","",INCR
else          SENDREQUEST(window,"","DECREMENT","",DECREME

/* Finished */
return;
}
/* ----- */
/* Handle server responding to INCREMENT */
/* ----- */

```

```

function INCREMENT_response(strFunction,strRequest,strPayload,objObject
{
    /* Handle a fatal error in the server function */
    if (flagFatalError) { alert(strFatalMessage); SETBUSY(false); return; }

    /* Otherwise display the result to the user */
    NUMBER_Tag.value = GetN("NUMBER").toString();
    MESSAGE_Tag.innerText = "Increment operation completed normally.";
    MESSAGE_Tag.style.backgroundColor = "yellow";
    SETBUSY(false);

    /* Finished */
    return;
}
/* ----- */
/* Handle server responding to DECREMENT */
/* ----- */
function DECREMENT_response(strFunction,strRequest,strPayload,objObject
{
    /* Handle a fatal error in the server function */
    if (flagFatalError) { alert(strFatalMessage); SETBUSY(false); return; }

    /* Otherwise display the result to the user */
    NUMBER_Tag.value = GetN("NUMBER").toString();
    MESSAGE_Tag.innerText = "Decrement operation completed normally.";
    MESSAGE_Tag.style.backgroundColor = "aqua";
    SETBUSY(false);

    /* Finished */
    return;
}
</script>

</HEAD>
<BODY id='HEAD_Tag'>
<br/><br/><br/>
<P align='center'>Hello World 2</p>
<span> Input Number ==&gt; </span><input id='NUMBER_Tag' type='text'>
<span> then click a button ==&gt; </span>

```

```
<input id='BUTTON1_Tag' type='button' value='Increment' onclick='BUTTON1_Click'>  
<input id='BUTTON2_Tag' type='button' value='Decrement' onclick='BUTTON2_Click'>  
<br/><br/><div id='MESSAGE_Tag' align='center'></div>  
</BODY>  
</HTML>
```

AJAX Function AJAXEX2

- * This is just a simple RDML function.
- * It is NOT a WAM function.
- * It is driven by the virtual clipboard

Function Options(*DIRECT *HEAVYUSAGE)

- * Define local fields

Define Field(#REQUEST) Reffld(#STD_OBJ)

Define Field(#NUMBER) Type(*DEC) Length(30) Decimals(9)

Define Field(#MSGDTA) Type(*CHAR) Length(132)

- * Get the action the Javascript requested in paramter 3
- * when it executed this ...
- * SENDREQUEST(window,"","INCREMENT",INCREMENT_response);
- * or this
- * SENDREQUEST(window,"","DECREMENT",DECREMENT_response);

Execute Subroutine(GETA) With_Parms(SYSTEM REQUEST 1 #REQUEST)

- * Now switch on the requested action

Case Of_Field(#REQUEST)

- * Get NUMBER from the clipboard, increment and put back

```
When Value_Is('= INCREMENT')
Execute Subroutine(GETN) With_Parms(NUMBER ' ' 1 #NUMBER)
Change Field(#NUMBER) To('#NUMBER + 1')
Execute Subroutine(PUTN) With_Parms(NUMBER ' ' 1 #NUMBER)
```

* Get NUMBER from the clipboard, decrement and put back

```
When Value_Is('= DECREMENT')
Execute Subroutine(GETN) With_Parms(NUMBER ' ' 1 #NUMBER)
Change Field(#NUMBER) To('#NUMBER - 1')
Execute Subroutine(PUTN) With_Parms(NUMBER ' ' 1 #NUMBER)
```

Otherwise

```
Use Builtin(BCONCAT) With_Args('Unknown request' #REQUEST 'received
Abort Msgid(DCM9899) Msgf(DC@M01) Msgdta(#MSGDTA)
Endcase
```

* Finished

Return

```
* =====
* Rationalized subroutines for virtual clipboard access
* =====
```

```
Subroutine Name(PUTA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST))
Define Field(#NP2) Type(*CHAR) Length(28)
Define Field(#NP3) Type(*CHAR) Length(24)
Define Field(#INST) Type(*DEC) Length(7) Decimals(0)
Define Field(#AVAL) Type(*CHAR) Length(256)
Use Builtin(VF_SAVEAVALUE) With_Args(#AVAL AJAX #NP2 #NP3 #INST)
Endroutine
```

```
Subroutine Name(PUTN) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST))
Define Field(#NVAL) Type(*DEC) Length(30) Decimals(9)
Use Builtin(VF_SAVENVALUE) With_Args(#NVAL AJAX #NP2 #NP3 #INST)
Endroutine
```

```
Subroutine Name(GETA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST))
```

Use Builtin(VF_RESTOREAVALUE) With_Args(' ' AJAX #NP2 #NP3 #INST
Endroutine

Subroutine Name(GETN) Parm((#NP2 *RECEIVED) (#NP3 *RECEIVED) (
Use Builtin(VF_RESTORENVALUE) With_Args(0 AJAX #NP2 #NP3 #INST
Endroutine

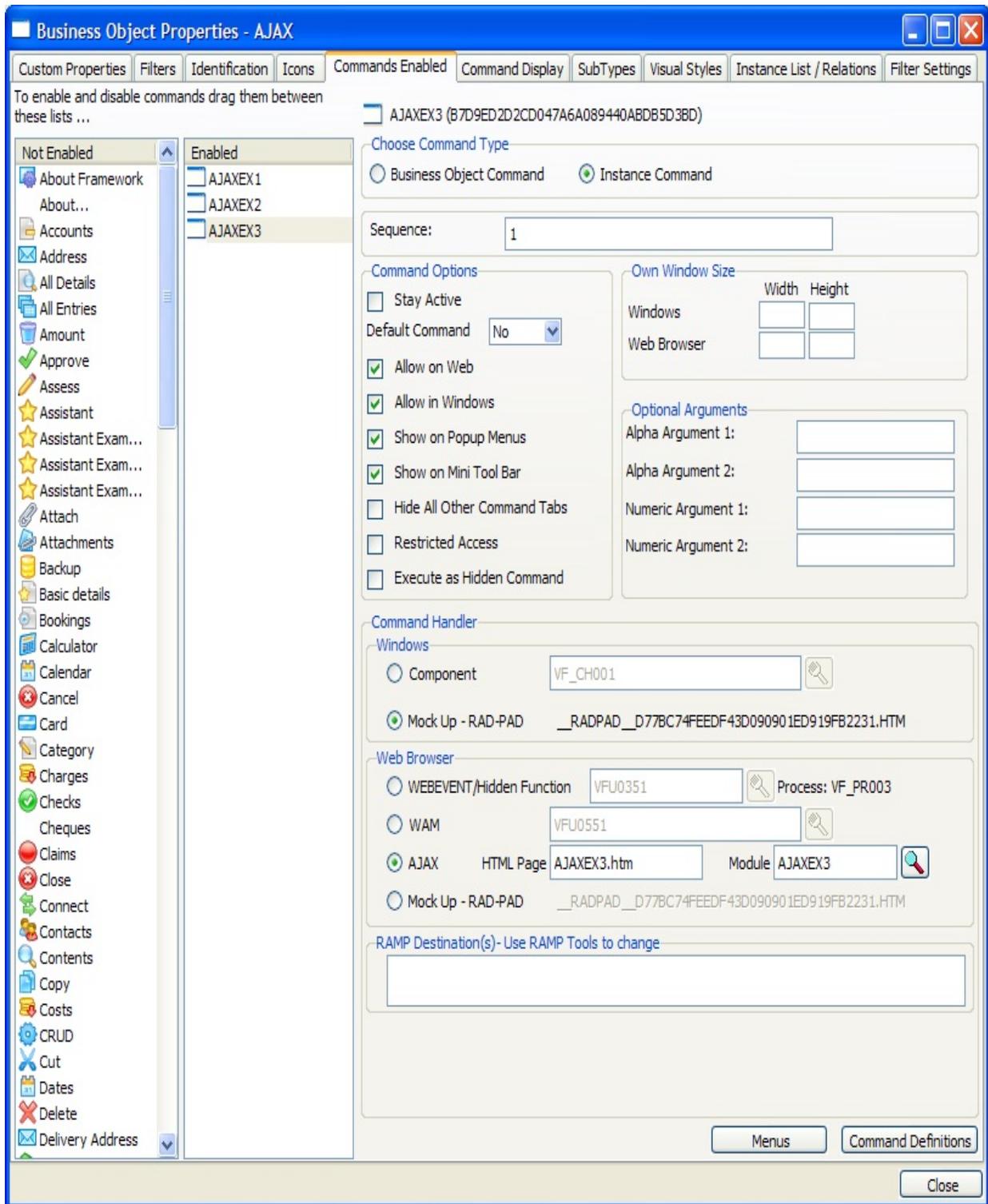
Exercise 3: AJAXEX3 – Using Asynchronous Processing

[AJAX Page AJAXEX3.HTM](#)

[AJAX Function AJAXEX3](#)

The Steps

- Copy and paste the DHTML-JavaScript code in [AJAX Page AJAXEX3.HTM](#) using NOTEPAD and save it into your Framework private working folder as an AJAX page file named AJAXEX3.HTM.
- Copy and paste the AJAX function code in [AJAX Function AJAXEX3](#) into an **RDMLX** function named AJAXEX3 into your VL-IDE. Compile RDML function AJAXEX3 so that it is executable on your web server.
- In business object AJAX, snap AJAX page file AJAXEX3.HTM and AJAX function AJAXEX3 into your Framework as the command handler to be associated with command AJAXEX3 like this:



Save, restart and execute your Framework in a web browser. Select application 'AJAX Examples' and then select business object AJAX. This should execute the default command AJAXEX1. Click on the AJAXEX3 tab to execute the

AJAXEX3 command:

High Volume Update

Name	Address	Zip Code	Status
Fred Bloggs	121 Smith St	32627	Loaded from Server
Mark Bloggs	121 Smith St	32627	Loaded from Server
Bill Bloggs	121 Smith St	32627	Loaded from Server
John Doe	11 Jones St	40210	Loaded from Server
Susan Doe	11 Jones St	40210	Loaded from Server
Junior Doe	11 Jones St	40210	Loaded from Server
John Hancock	15 Wash Road	40314	Loaded from Server
Helen Hancock	15 Wash Road	6314	Loaded from Server
Fred Hancock	15 Wash Road	60314	Loaded from Server
Mark Smith	151 Teller Avenue	60315	Loaded from Server

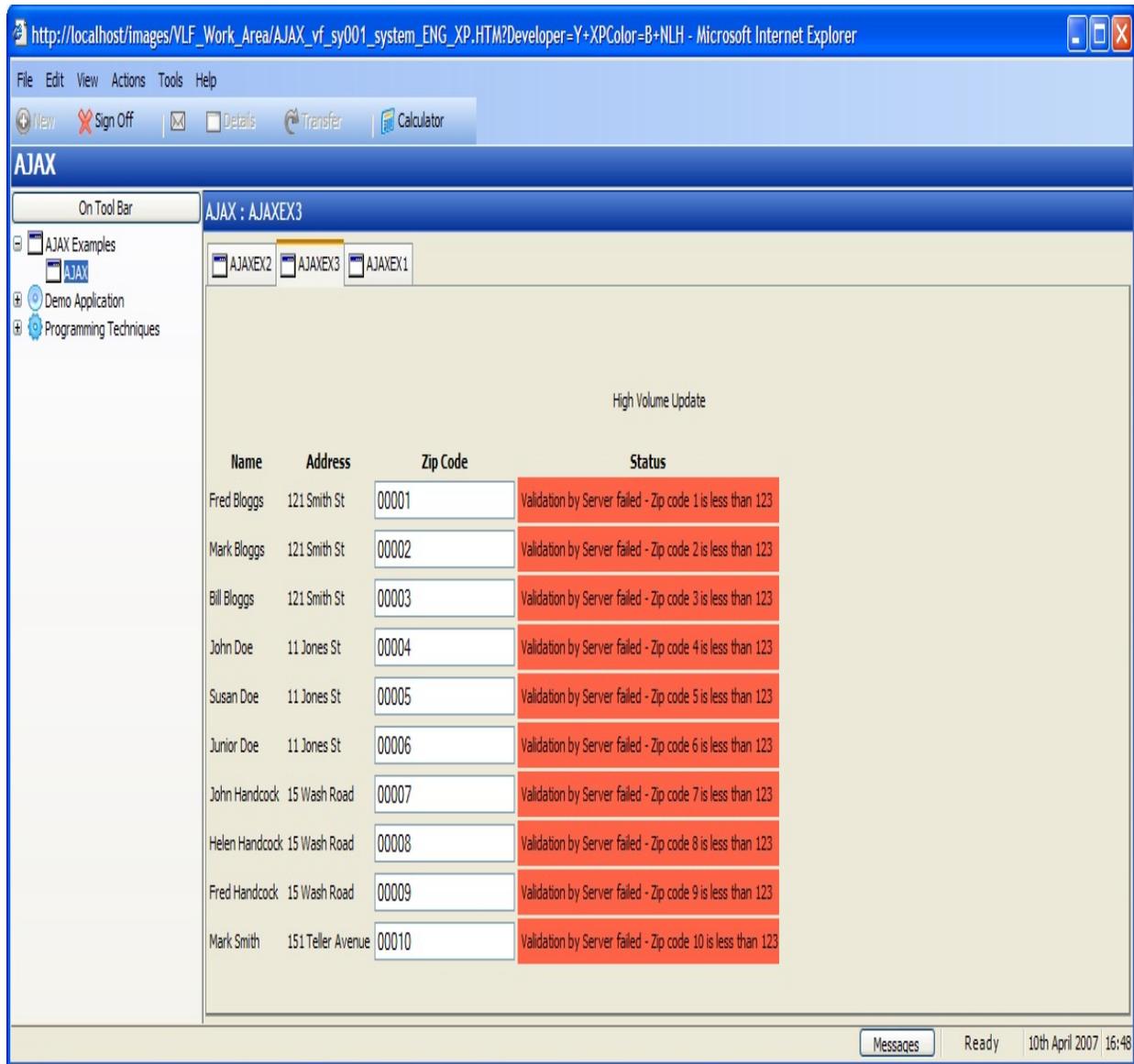
This AJAX example is designed to demonstrate the asynchronous nature of AJAX processing.

It is updating the zip codes associated with people. As you change zip codes and press Enter or tab your change will be sent to the server for processing. While this is happening you can update other zip codes.

Try this sequence:

- Click on the first zip code and select it.

Then type “1”, Tab ->, “2”, Tab ->, “3”, Tab ->, etc as fast as you can. The display should look like this:



As you make each change and press the tab -> key, your change is sent to the server for immediate validation. You can then keep making changes while the server handles this request.

In a real application you are more likely to be entering order lines (say) and the changes you make are updated immediately into the server database.

You should review the JavaScript/DHTML code in `AJAXEX3.HTM` and the RDML code in `AJAX` function `AJAXEX3` until you understand the basics of

what it is doing and how it is doing it.

AJAX Page AJAXEX3.HTM

```
<HTML>
<HEAD id='HEAD_Tag'>
<link rel='stylesheet' type='text/css' href='vf_vs001.css' >
<script>
/* =====
/* ===== Handle Page Initialization =====
/* =====
function VF_AJAX_Initialize()
{
  /* Insert the variable style sheet (ie: XP, WIN, WEB style) into this page */
  {
    var objLink = document.createElement("LINK");
    objLink.rel = "stylesheet"; objLink.type = "text/css";
objLink.href = STYLESHEET();
    document.getElementById("HEAD_Tag").insertAdjacentElement("afterBe
  }

  /* Finished */
  return;
}
/* =====
/* ===== Handle Page Execution =====
/* =====
function VF_AJAX_Execute()
{
  /* Request that the server makes up 10 example name/address/zipcode details
  SENDREQUEST(window,"","LOAD10","",LOAD10_response);
  /* Finished */
  return;
}
/* =====
/* ===== Handle Page Termination =====
/* =====
```

```

function VF_AJAX_Terminate()
{
    return;
}
/* ----- */
/* Rationalized virtual clipboard access to name space AJAX */
/* ----- */
function Put(val,np2,np3,inst) {AVSAVEVALUE(val,"AJAX",np2,np3,inst);}
function GetA(np2,np3,inst)  {return(AVRESTOREAVALUE("", "AJAX",np2
function GetN(np2,np3,inst)  {return(AVRESTORENVALUE(0,"AJAX",np2
/* ----- */
/* Handle server responding to LOAD10 */
/* ----- */
function LOAD10_response(strFunction,strRequest,strPayload,objObject,flagF
{
    var inst = 0;
    /* Handle a fatal error in the server function */
    if (flagFatalError) { alert(strFatalMessage); SETBUSY(false); return; }

    /* Otherwise load the table with the 10 results and display to the user */
    for (inst = 1; inst <= 10; inst++)
    {
        var objTR      = TBODY_Tag.rows(inst - 1);
        var objTDName  = objTR.children(0);
        var objTDAddress = objTR.children(1);
        var objINZipCode = objTR.children(2).children(0);
        var objTDStatus = objTR.children(3);

        objTDName.innerText  = GetA("EX3","NAME",inst);
        objTDAddress.innerText = GetA("EX3","ADDRESS",inst);
        objINZipCode.value    = GetN("EX3","ZIPCODE",inst).toString();
        objINZipCode.__validatedvalue = objINZipCode.value;
        objTDStatus.innerText = " Loaded from Server ";
        objTDStatus.style.backgroundColor = "palegreen";
    }

    /* Hide the message panel and show the table panel */

```

```

DIV_Message.style.visibility = "hidden"; DIV_Message.style.display = "no
DIV_Table.style.visibility = "visible"; DIV_Table.style.display = "inline"

/* Drop busy status */
SETBUSY(false);

/* Finished */
return;
}
/* ----- */
/* Handle key press in TBODY and tabbing to next input zip code */
/* ----- */
function NextZipCode(objINZipCode)
{
    var intNext = parseInt(objINZipCode.__Instance,10) + 1;
    if (intNext > 10) intNext = 1;
    var objNextINPUT = document.getElementById("IN" + intNext.toString());
    if (objNextINPUT != null) objNextINPUT.focus();
}

function TBODY_onkeydown()
{
    /* Ignore if not valid, or not enter or ab key */
    if ((event.keyCode != 13) && (event.keyCode != 9)) return;
    if (typeof(event.srcElement.__validatedvalue) == "undefined") return;

    /* Extract the INPUT field involved, ignore if not zip code or not changed */
    {
        var objINZipCode = event.srcElement;
        /* If unchanged do nothing */
        if (objINZipCode.__validatedvalue == objINZipCode.value )
        {
            if (event.keyCode == 13) NextZipCode(objINZipCode);
        }
    }
    else
    {
        while (objINZipCode.value.length < 5) objINZipCode.value = "0" + objIN
        var floatNumber = parseFloat(objINZipCode.value);
        /* var intInstance = parseInt(objINZipCode.__Instance,10); */

```

```

var intInstance = parseInt(objINZipCode.getAttribute("__Instance"),10);

if (isNaN(floatNumber)) { alert("Invalid number entered"); return; }
/* reaching here means a zip code that has been changed, so send validation
{
    var objTDStatus = objINZipCode.parentElement.nextSibling;
objTDStatus.innerHTML = " Validating at Server ";
    objTDStatus.style.backgroundColor = "gold";
}

objINZipCode.contentEditable = false;
Put(floatNumber,"EX3","ZIPCODE",intInstance);
SENDREQUEST(window,"","VALIDATE",intInstance.toString(),VALIDATE_
if (event.keyCode == 13) NextZipCode(objINZipCode);
}
}

/* Finished */
return;
}
/* ----- */
/* Handle server responding to VALIDATE */
/* ----- */
function VALIDATE_response(strFunction,strRequest,strPayload,objINZipCode)
{
    var objTDStatus = objINZipCode.parentElement.nextSibling;

    /* Handle a fatal error in the server function */
    if (flagFatalError) { alert(strFatalMessage); return; }

    objINZipCode.contentEditable = true;

    if (GetA("EX3","RESPONSE") == "OK")
    {
        objINZipCode.__validatedvalue = objINZipCode.value;
        objTDStatus.innerHTML = " Validated and Updated by Server ";
        objTDStatus.style.backgroundColor = "springgreen";
    }
    else
    {

```

```
objTDStatus.innerText = " Validation by Server failed - " + GetA("EX3","M
objTDStatus.style.backgroundColor = "tomato";
}

/* Zap the object reference */
objINZipCode = null;
}

</script>

</HEAD>
<BODY id='BODY_Tag'>
<br/><br/><br/>
<P align='center'>High Volume Update</p>
<div id='DIV_Message'>
<P align='center'>Loading ... please wait</p>
</div>
<div id='DIV_Table' style='visibility:hidden;display:none;'>
<table>
<thead>
<tr>
<th>Name</th><th>Address</th><th>Zip Code</th><th>Status</th></tr>
</thead>
<tbody id='TBODY_Tag' onkeydown='TBODY_onkeydown();'>
<tr><td></td> <td> </td> <td>
<input id='IN1' type='text' maxlength=5' __Instance=1></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN2' type='text' maxlength=5' __Instance=2></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN3' type='text' maxlength=5' __Instance=3></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN4' type='text' maxlength=5' __Instance=4></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN5' type='text' maxlength=5' __Instance=5></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN6' type='text' maxlength=5' __Instance=6></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN7' type='text' maxlength=5' __Instance=7></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
```

```

<input id='IN8' type='text' maxlength=5' __Instance=8></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN9' type='text' maxlength=5' __Instance=9></td><td> </td></tr>
<tr><td></td> <td> </td> <td>
<input id='IN10' type='text' maxlength=5' __Instance=10></td><td> </td>
</tr>
</tbody>
</table>
</div>
</BODY>
</HTML>

```

AJAX Function AJAXEX3

- * This is just a simple RDMLX function.
- * It is NOT a WAM.
- * It is driven by the virtual clipboard

Function Options(*DIRECT *HEAVYUSAGE)

- * Define local fields

Define Field(#REQUEST) Reffld(#STD_OBJ)

Define Field(#PAYLOAD) Reffld(#STD_TEXTL)

Define Field(#INSTANCE) Type(*DEC) Length(7) Decimals(0)

- * Get the action and payload Javascript supplied

Execute Subroutine(GETA) With_Parms(SYSTEM REQUEST 1 #REQUEST)

Execute Subroutine(GETA) With_Parms(SYSTEM PAYLOAD 1 #PAYLOAD)

- * Now switch on the requested action

Case Of_Field(#REQUEST)

- * Load 10 sample entries

When Value_Is(= LOAD10)

Execute Subroutine(Load10)

* Validate a zip code

When Value_Is(= VALIDATE)

Execute Subroutine(VALIDATE)

* Handle a bad request

Otherwise

Abort Msgid(DCM9899) Msgf(DC@M01) Msgdta(('Unknown request ' + #RE

Endcase

* Finished

Return

* =====

* Request handling subroutines

* =====

Subroutine Name(Validate)

* The payload contains the instance number of the zip code

* that is to be validated by this request, so get it as a number

#Instance := #Payload.AsNumber()

* Now get the zip code number associated with the instance

Execute Subroutine(GetN) With_Parms(EX3 ZipCode #Instance #Std_Num)

* Now validate the zip code and give response codes and error messages

Case (#Std_Num)

When (< 123)

```
Execute Subroutine(PutA) With_Parms(EX3 Response 1 ER)
Execute Subroutine(PutA) With_Parms(EX3 Message 1 ('Zip code ' + #Std_Nu
```

```
When (> 77777)
```

```
Execute Subroutine(PutA) With_Parms(EX3 Response 1 ER)
Execute Subroutine(PutA) With_Parms(EX3 Message 1 ('Zip code ' + #Std_Nu
```

```
Otherwise
```

```
Execute Subroutine(PutA) With_Parms(EX3 Response 1 OK)
```

```
Endcase
```

```
Endroutine
```

```
Subroutine Name(Load10)
```

```
#Instance := 0
```

```
Execute Subroutine(Add) With_Parms('Fred Bloggs' '121 Smith St' 32627)
```

```
Execute Subroutine(Add) With_Parms('Mark Bloggs' '121 Smith St' 32627)
```

```
Execute Subroutine(Add) With_Parms('Bill Bloggs' '121 Smith St' 32627)
```

```
Execute Subroutine(Add) With_Parms('John Doe' '11 Jones St' 40210)
```

```
Execute Subroutine(Add) With_Parms('Susan Doe' '11 Jones St' 40210)
```

```
Execute Subroutine(Add) With_Parms('Junior Doe' '11 Jones St' 40210)
```

```
Execute Subroutine(Add) With_Parms('John Handcock' '15 Wash Road' 40314)
```

```
Execute Subroutine(Add) With_Parms('Helen Handcock' '15 Wash Road' 6314)
```

```
Execute Subroutine(Add) With_Parms('Fred Handcock' '15 Wash Road' 60314)
```

```
Execute Subroutine(Add) With_Parms('Mark Smith' '151 Teller Avenue' 60315)
```

```
Endroutine
```

```
Subroutine Name(Add) Parms((#Std_text *received) (#Std_textl *received) (#
```

```
#Instance += 1
```

```
Execute Subroutine(PutA) With_Parms(EX3 Name #Instance #Std_Text)
```

```
Execute Subroutine(PutA) With_Parms(EX3 Address #Instance #Std_Textl)
```

```
Execute Subroutine(PutN) With_Parms(EX3 ZipCode #Instance #Std_Num)
```

```
Endroutine
```

```
* =====
```

```
* Rationalized subroutines for virtual clipboard access
```

* =====

* =====

* Rationalized subroutines for virtual clipboard access

* =====

```
Subroutine Name(PUTA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST *RECEIVED))
Define Field(#NP2) Type(*CHAR) Length(28)
Define Field(#NP3) Type(*CHAR) Length(24)
Define Field(#INST) Type(*DEC) Length(7) Decimals(0)
Define Field(#AVAL) Type(*CHAR) Length(256)
Use Builtin(VF_SAVEAVALUE) With_Args(#AVAL AJAX #NP2 #NP3 #INST)
Endroutine
```

```
Subroutine Name(PUTN) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST *RECEIVED))
Define Field(#NVAL) Type(*DEC) Length(30) Decimals(9)
Use Builtin(VF_SAVENVALUE) With_Args(#NVAL AJAX #NP2 #NP3 #INST)
Endroutine
```

```
Subroutine Name(GETA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST *RECEIVED))
Use Builtin(VF_RESTOREAVALUE) With_Args(' ' AJAX #NP2 #NP3 #INST)
Endroutine
```

```
Subroutine Name(GETN) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (#INST *RECEIVED))
Use Builtin(VF_RESTORENVALUE) With_Args(0 AJAX #NP2 #NP3 #INST)
Endroutine
```

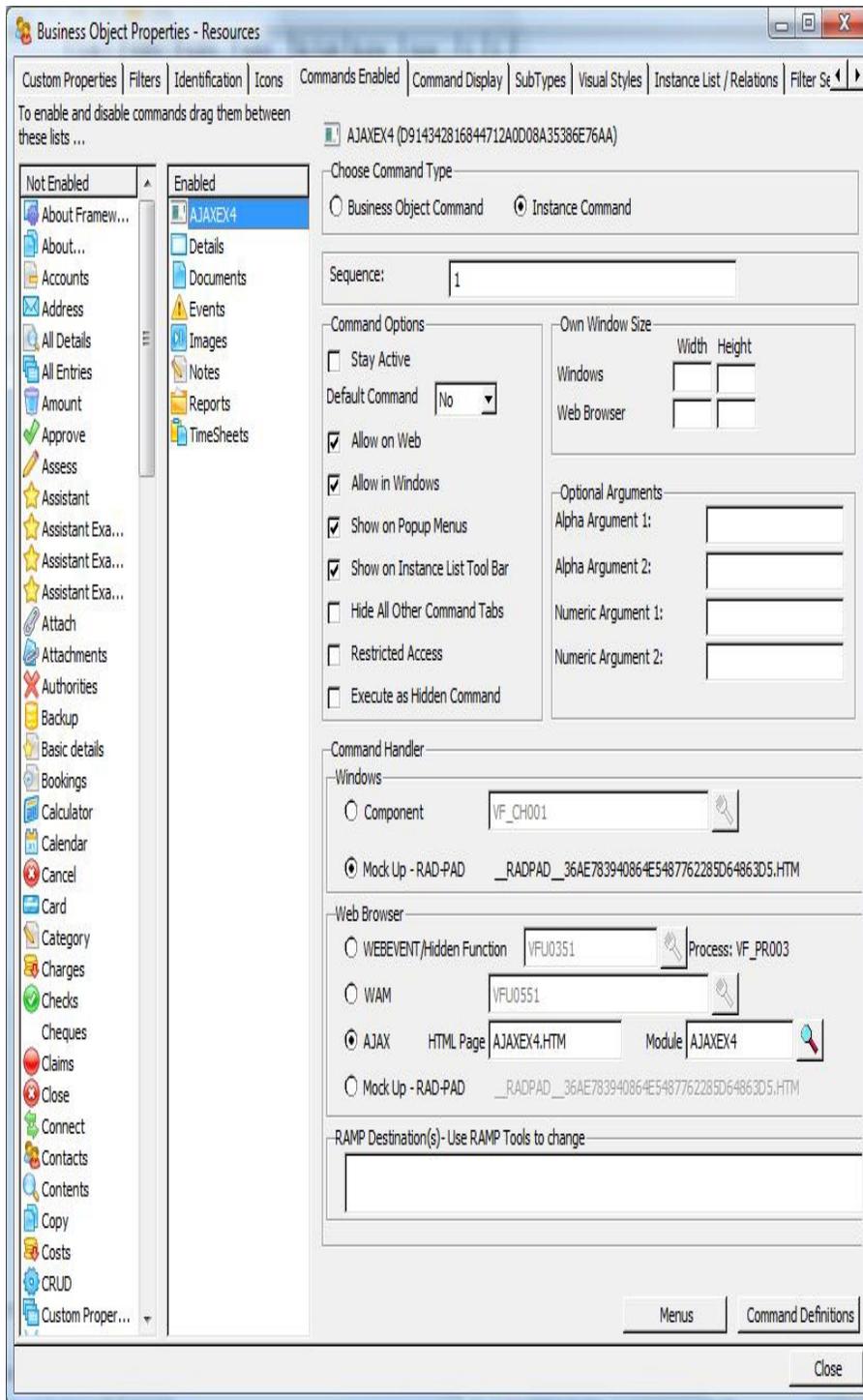
Exercise 4: AJAXEX4 – Using the Instance List

[AJAX Page AJAXEX4.HTM](#)

[AJAX Function AJAXEX4](#)

The Steps

- Copy and paste the DHTML-JavaScript code in [AJAX Page AJAXEX4.HTM](#) using NOTEPAD and save it into your Framework private working folder as an AJAX page file named AJAXEX4.HTM.
- Copy and paste the AJAX function code in [AJAX Function AJAXEX4](#) into an RDML function named AJAXEX4 into your VL-IDE. Compile RDML function AJAXEX4 so that it is executable on your web server.
- In the shipped demonstration business object “Resources” add command AJAXEX4 as a new instance level command (like Details, Transfer, etc).
- Snap AJAXEX4.HTM and AJAXEX4 into as the handlers associated with command AJAXEX4 like this:



Save, restart and execute your Framework in a web browser. Select the “Resources” business object and build a list of employees. Select one. This should display the default “Details” command. Switch to the AJAXEX4 tab and

you should see the details of the employee displayed like this (in VLF.WEB):

The screenshot shows a web browser window with the following elements:

- Browser Address Bar:** `http://localhost/images/rd1_private/VF_SY001_SYSTEM_LASTSHIPPED_ENG_XP.HTM?Developer=Y+NLHostNa - Windows Internet Explorer`
- Search Bar:** Contains the text "br". Search options include "Search by Name or Number", "Search by Phone Number", and "Search by Skill Code".
- Table of Employees:**

Name	Code / Id	Address Line 1	Address Line 2	Address Line 3	Zip Code	Business Phone	Home Phone	Department	Section
BROWN,VERONICA	A0070	12 Railway Street	Baulkham Hills	NSW Australia		2153 (02) 9647 2788	(02) 9609 4627	INF	DV
WOODS,BRADLEY	A1015	59 Darley Road,	BEXLEY,	NSW,		2030 789 4562	450 1226	AUD	02
MRS BRICK,GILL	A1404	22 Moton Street	Marrickville	NSW		2090 324 444	3343 333	ADM	01
BROWN,FREDDY	A3564	121 SMITH STREET	Newtown			2153 (02) 456-5678	(02) 567-6758	ADM	04

Resource: AJAXEX4 (WOODS,BRADLEY - A1015)

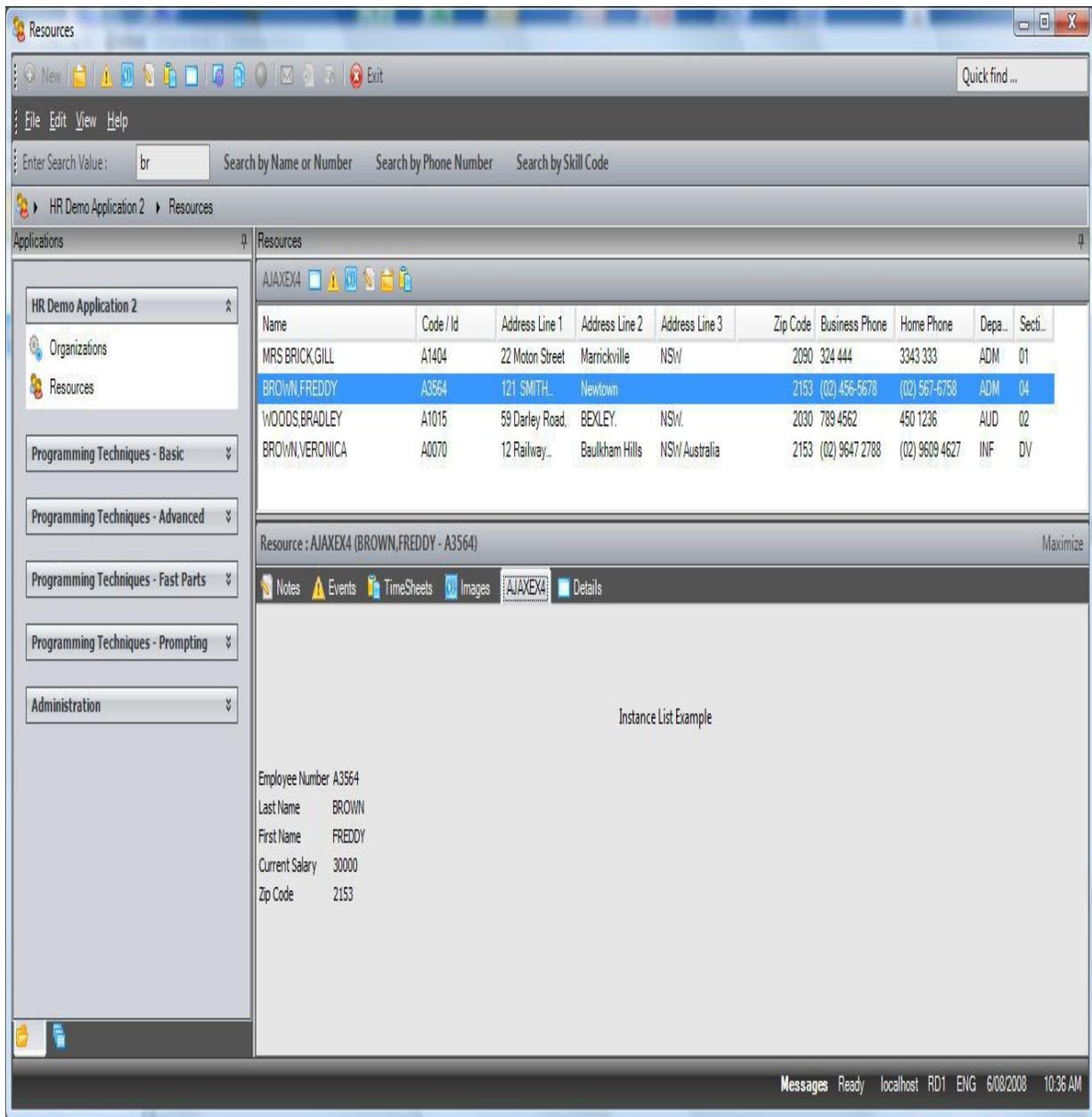
Navigation tabs: Notes, Events, TimeSheets, Images, AJAXEX4, Details

Instance List Example

Employee Number A1015
Last Name WOODS
First Name BRADLEY
Current Salary 313000.04
Zip Code 2030

System Status: System Available | Local intranet | Protected Mode: Off | 100% | 6th August 2008 10:34

Or this (VLF.NET):



This AJAX example is designed to demonstrate how business object instance list details are passed into AJAX pages.

They are passed into the VF AJAX_Execute function as a JavaScript instance list object which contains these properties and arrays:

Name	Type
strVisualID1	String. Visual ID1
strVisualID2	String. Visual ID2.

strBusinessObjectType	String. Business Object Type.
arraystrAKeyN	Array[5] String. AKey values.
arraynumNKeyN	Array[5] numbers. NKey values.
intAColumnCount	Count of additional alpha columns.
arraystrAColumn	Array of additional alpha columns.
intNColumnCount	Count of additional numeric columns.
arraynumNColumn	Array of additional numeric columns.
<all others>	There may be other values and properties in this object. You should NOT reference them in your code to avoid future version incompatibilities and/or failures.

Note 1: Some of these values may be blank padded and needed to be trimmed for presentation.

Note 2: The Framework-AJAX facility is primarily designed to be used in full screen command handlers that largely act as on their own, as demonstrated in the preceding exercises 1 through 3.

AJAX Page AJAXEX4.HTM

```

<HTML>
<HEAD id='HEAD_Tag'>
<link rel='stylesheet' type='text/css' href='vf_vs001.css' >
<script>
/* =====
/* ===== Handle Page Initialization =====
/* =====
function VF_AJAX_Initialize()
{

```

```

/* Insert the variable style sheet (ie: XP, WIN, WEB style) into this page */
{
    var objLink = document.createElement("LINK");
    objLink.rel = "stylesheet"; objLink.type = "text/css";
objLink.href = STYLESHEET();
    document.getElementById("HEAD_Tag").insertAdjacentElement("afterBe
}

/* Finished */
return;
}
/* =====
/* ===== Handle Page Execution =====
/* =====
function VF_AJAX_Execute(objInstance)
{
    /* Put the current instance list entry EMPNO onto the clipboard          */
    /* The employee number is in AKey number 3, which is Javascript array entry
Put(objInstance.arraystrAKeyN[2],"EX4","EMPNO");

    /* Request that the server do a FETCH operation using the employee number
SENDREQUEST(window,"","FETCH","",FETCH_response);

    /* Finished */
    return;
}
/* =====
/* ===== Handle Page Termination =====
/* =====
function VF_AJAX_Terminate()
{
    return;
}
/* ----- */
/* Rationalized virtual clipboard access to name space AJAX */
/* ----- */
function Put(val,np2,np3,inst) {AVSAVEVALUE(val,"AJAX",np2,np3,inst);}
function GetA(np2,np3,inst)   {return(AVRESTOREAVALUE("", "AJAX",np2
function GetN(np2,np3,inst)   {return(AVRESTORENVALUE(0,"AJAX",np2

```

```

/* ----- */
/* Handle server responding to LOAD10 */
/* ----- */
function FETCH_response(strFunction,strRequest,strPayload,objObject,flagFa
{
/* Handle a fatal error */
if (flagFatalError) { alert(strFatalMessage); SETBUSY(false); return; }

/* Put the employee number out onto the screen */
Vis_EMPNO.innerText = GetA("EX4","EMPNO");

/* Get the IO Status and handle found and not found */

if (GetA("EX4","IO_STS") == "OK") /* Employee was found */
{
Vis_SURNAME.innerText = GetA("EX4","SURNAME");
Vis_GIVENAME.innerText = GetA("EX4","GIVENAME");
Vis_SALARY.innerText = GetN("EX4","SALARY").toString();
Vis_POSTCODE.innerText = GetN("EX4","POSTCODE").toString();
DIV_Table.style.visibility = "visible"; DIV_Table.style.display = "inline";
DIV_Message.style.visibility = "hidden"; DIV_Message.style.display = "no
}
else /* Employee was not found */
{
DIV_Message.children(0).innerText = "Employee number " + Vis_EMPNO
DIV_Message.style.visibility = "visible"; DIV_Message.style.display = "inl
DIV_Table.style.visibility = "hidden"; DIV_Table.style.display = "none";
}

/* Drop busy status */
SETBUSY(false);

/* Finished */
return;
}
</script>

</HEAD>
<BODY id='BODY_Tag'>

```

```

<br/><br/><br/>
<P align='center'>Instance List Example</p>
<div id='DIV_Message'>
<P align='center'>Loading ... please wait</p>
</div>
<div id='DIV_Table' style='visibility:hidden;display:none;'>
<table>
<tr><td>Employee Number</td><td id='Vis_EMPNO'></td></tr>
<tr><td>Last Name</td><td id='Vis_SURNAME'></td></tr>
<tr><td>First Name</td><td id='Vis_GIVENAME'></td></tr>
<tr><td>Current Salary</td><td id='Vis_SALARY'></td></tr>
<tr><td>Zip Code</td><td id='Vis_POSTCODE'></td></tr>
</table>
</div>
</BODY>
</HTML>

```

AJAX Function AJAXEX4

- * This is just a simple RDML function.
- * It is NOT a WAM.
- * It is driven by the virtual clipboard

Function Options(*DIRECT *HEAVYUSAGE)

- * Define local fields

Define Field(#REQUEST) Reffld(#STD_OBJ)

Define Field(#MSGDTA) Type(*CHAR) Length(132)

- * Get the action the Javascript requested

Execute Subroutine(GETA) With_Parms(SYSTEM REQUEST 1 #REQUEST)

- * Now switch on the requested action

Case Of_Field(#REQUEST)

* Get details of an employee

When Value_Is('= FETCH')

Execute Subroutine(FETCH)

Otherwise

Use Builtin(BCONCAT) With_Args('Unknown request' #REQUEST 'received

Abort Msgid(DCM9899) Msgf(DC@M01) Msgdta(#MSGDTA)

Endcase

* Finished

Return

* Get details of an employee

Subroutine Name(FETCH)

* Get the requested employee number

Execute Subroutine(GETA) With_Parms(EX4 EMPNO 1 #EMPNO)

* Fetch the details

Fetch Fields(#Surname #GiveName #PostCode #Salary) From_File(Pslmst) W

* Send back the IO status + the details

Execute Subroutine(PUTA) With_Parms(EX4 IO_STS 1 #IO\$STS)

Execute Subroutine(PUTA) With_Parms(EX4 SURNAME 1 #SURName)

Execute Subroutine(PUTA) With_Parms(EX4 GIVENAME 1 #GIVENAME)

Execute Subroutine(PUTN) With_Parms(EX4 POSTCODE 1 #POSTCODE)

Execute Subroutine(PUTN) With_Parms(EX4 SALARY 1 #SALARY)

Endroutine

* =====

* Rationalized subroutines for virtual clipboard access

* =====

```
Subroutine Name(PUTA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (:
Define Field(#NP2) Type(*CHAR) Length(28)
Define Field(#NP3) Type(*CHAR) Length(24)
Define Field(#INST) Type(*DEC) Length(7) Decimals(0)
Define Field(#AVAL) Type(*CHAR) Length(256)
Use Builtin(VF_SAVEAVALUE) With_Args(#AVAL AJAX #NP2 #NP3 #INS
Endroutine
```

```
Subroutine Name(PUTN) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (
Define Field(#NVAL) Type(*DEC) Length(30) Decimals(9)
Use Builtin(VF_SAVENVALUE) With_Args(#NVAL AJAX #NP2 #NP3 #INS
Endroutine
```

```
Subroutine Name(GETA) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (
Use Builtin(VF_RESTOREAVALUE) With_Args(' ' AJAX #NP2 #NP3 #INST
Endroutine
```

```
Subroutine Name(GETN) Parms((#NP2 *RECEIVED) (#NP3 *RECEIVED) (
Use Builtin(VF_RESTORENVALUE) With_Args(0 AJAX #NP2 #NP3 #INST
Endroutine
```

Tracing

[Tracing in Your Client Side AJAX Pages](#)

[Tracing in Your Server Side AJAX Functions](#)

Tracing in Your Client Side AJAX Pages

You can add tracing to your AJAX pages by using the supplied AVTRACE function.

The information you trace appears in the trace window when you execute the Framework in trace mode. The AVTRACE function is defined like this:

```
AVTRACE("Example1","Search Button Clicked","");
```

Here's an example of an AJAX page tracing data when the user clicks a search button:

```
function BUTTON_Search()
{
    /* Ignore if already busy */
    if (BUSY()) return;

    /* Make busy now */
    SETBUSY(true);

    /* Put surname and given name input by user onto virtual clipboard */
    PutA(SURNAME_Search.value,"SURNAME");
    PutA(GIVENAME_Search.value,"GIVENAME");

    /* Trace the values that were put onto the clipboard */
    AVTRACE("BUTTON_Search","Surname value set",SURNAME_Search.va
    AVTRACE("BUTTON_Search","Given Name value set",GIVENAME_Search
    ...etc...
```

Tracing in Your Server Side AJAX Functions

You can add tracing to your AJAX functions by using the Framework VF_TRACENVALUE and VF_TRACEAVALUE built-in functions.

The information you trace appears in the trace window when you execute the Framework in trace mode. Here is an example of a AJAX function tracing a FETCH operation:

```
Subroutine Name(GET)

* Get the employee number supplied and trace it

Execute Subroutine(GETA) With_Parms(EMPNO ' ' 1 #EMPNO)

Use Builtin(VF_TraceAvalue) With_Args('GET received employee number' #E

Fetch Fields(#PSLMST) From_File(PSLMST) With_Key(#EMPNO) Issue_M

Execute Subroutine(PUTA) With_Parms(IO_STATUS ' ' 1 #IO$STS)

* If the record was found trace the salary and surname values retrieved

If_Status Is(*OKAY)
Use Builtin(VF_TraceNvalue) With_Args('GET request found salary' #Salary)
Use Builtin(VF_TraceAvalue) With_Args('GET request found surname' #Surname)

...etc...
```

VF_TRACEAVALUE and VF_TRACENVALUE

The VF_TRACEAVALUE and VF_TRACENVALUE Built-In functions are used to trace the logic flow and the values of variables in your browser command handlers and filters.

The values that are traced are added to the trace window that is displayed when Framework Web browser applications are executed in trace mode (by adding ? Trace=Y to the URL that is used to start the application or by checking the Trace Mode On option).

Generally trace Built-In functions may be left in your code as they are only activated when trace mode is turned on. Do not do this in code sections that are executed many times per interaction.

For use with

LANSA for i	YES
Visual LANSA for Windows	YES
Visual LANSA for UNIX/Linux	NO

Arguments

No	Type	Req/ Opt	Description	Min Len	Max Len	Min Dec	Max Dec
1	A	Req	Event Description	1	256		
2	A N	Opt	Alphanumeric Value 1 to trace Numeric Value 1 to trace	1 1	256 30	0	9
3	A N	Opt	Alphanumeric Value 2 to trace Numeric Value 2 to trace	1 1	256 30	0	9
4	A N	Opt	Alphanumeric Value 3 to trace Numeric Value 3 to trace	1 1	256 30	0	9

Return Values

None.

Technical Notes

The general usage rules apply to this Built-In function.

Examples

Example 1: This code fragment is seen in many applications generated by the Program Coding Assistant:

```
Use Builtin(VF_TRACEVALUE) With_Args('#VF_EVENT value is unknow
```

Example 2: Trace some alpha values:

```
Use Builtin(VF_TRACEAVALUE) With_Args('Department and section are' #I
```

Example 3: Trace some numeric values:

```
Use Builtin(VF_TRACENVALUE) With_Args('Salary=' #SALARY)
```

```
Use Builtin(VF_TRACENVALUE) With_Args('Zip Code=' #POSTCODE)
```

Messages and Errors

Messages Issued by AJAX Functions

Handling Fatal Errors in AJAX Functions

Messages Issued by AJAX Functions

By default, messages returned by your AJAX functions are collected by the Framework and routed into the Framework message area. This means that they will appear on the browser task bar and be visible when the messages button is used.

You can stop this happening.

When invoking the SENDREQUEST function pass the 8th parameter as false to turn off this behavior:

```
Eg: SENDREQUEST(window,"","LOAD10","",LOAD10_response,null,true,f
```

You can access the messages yourself by retrieving them from the AJAX virtual clipboard object like this example does:

```
var i = 0;
var intMESSAGECOUNT = AVRESTORENVALUE(0,"AJAX","SYSTEM

for (i = 1; i <= intMESSAGECOUNT; i++)
{
    Var strMessageText = AVRESTOREAVALUE ("","AJAX","SYSTEM",")
    alert(strMessageText);
}

AVSAVEVALUE(0,"AJAX","SYSTEM","MESSAGECOUNT");
```

Handling Fatal Errors in AJAX Functions

By default fatal errors in your server side AJAX functions are generally trapped and displayed by the Framework.

You can stop this from happening and handle the error display yourself.

When invoking the SENDREQUEST function pass the 7th parameter as false.

```
Eg: SENDREQUEST(window,"","LOAD10","",LOAD10_response,null,false)
```

This tells the Framework not display the fatal error details itself.

In all cases your JavaScript request handler is always called with parameters that indicate that a fatal error has occurred and a composite message is made up of all the trapped error messages.

```
function Resp(strFunction,strRequest,strPayload,objObject,flagErrorFatal,strFatalMessage)
{
/* Handle a fatal error in the server function */

if (flagFatalError)
{
    alert(strFatalMessage);
    SETBUSY(false);
    return;
}

/* Else process the response from the server */

... etc .....
```

Note: The fatal error flag refers to fatal errors in the server side AJAX function, not to fatal errors in your client side JavaScript, which you should trap and handle yourself.

Frequently Asked Questions

Are AJAX applications secure?

In general internet usage, AJAX applications are not always secure (depending up how they were designed). However, the Framework implementation is secure.

AJAX applications can only be used within Framework managed web sessions. They can be used over https connections.

How do I screen paint my AJAX pages?

AJAX pages are standard HTML documents which can be painted with many different tools, such as Microsoft FrontPage, etc.

However, as you start to do more advanced AJAX applications you will see that static screen painters are really not that useful when dealing with web pages where much of the content is dynamically created and updated by executing JavaScript AJAX code (this is also true of many Windows Rich Client screen painters as well).

Do I need to backup my AJAX pages?

Yes. It is very important that you do this or you will lose your work.

Do I need to backup my AJAX functions?

Yes, but not independently of your LANSAs repository.

Since AJAX functions are just LANSAs functions they will be backed up whenever you backup your LANSAs development environment.

Restrictions

Some restrictions that you should consider are:

- AJAX functions can be coded as RDML or RDMLX (depending your functional requirements and application design, RDML and RDMLX functions executing on an i5 server may exhibit different performance characteristics).
- The Framework manager object #VF_SW100 (typically named #AVFrameworkManager) can only be used in WAMs. The current AJAX implementation does not support use of the Framework manager.
- The virtual clipboard is the only means of exchanging information between AJAX pages and AJAX functions. Alpha values placed on it are limited to 256 characters. Numeric values are limited to a maximum precision of 21 significant digits and 9 decimals. If you need longer alpha or numeric fields you may need to break you information up in the AJAX page pieces and reassemble it in the AJAX function.
- The operation, maintenance and support of the JavaScript and HTML you put into your AJAX pages is entirely your responsibility, both now and in the future.
- When coding your AJAX pages the Framework's internal JavaScript and DHTML model is exposed to you. You should not use any of its methods, properties or events, nor alter its behavior in any way, unless the technique you use is specifically documented by LANSa for public usage. Failure to observe this rule may result in the introduction of incompatibilities with future versions of the Framework, voiding or limitation of any maintenance contract you may have in place for the Framework and being charged for problem resolutions that are traced back to any such usage or modifications.

Recommendations

- As you develop more AJAX pages you will probably start to build up a library of common JavaScript functions. These are best externalized in a separate .JS and shared by all your AJAX pages, rather than repeated.
- Set up some naming and usage standards for the content of the virtual clipboard. This will make life easier for other developers, encourage reuse and minimize information bloat. For example:

Major Name Space: **AJAX** (virtual clipboard name part 1)

Minor Name Space: **CUSTOMER** (virtual clipboard name part 2)

Object Name (Name Part 3)	Description	Type
NAME	Current Customer Name	String – max 40
ZIPCODE	Current Customer Zip Code	Number – 6 digits

In effect, you have AJAX.CUSTOMER.NAME and AJAX.CUSTOMER.ZIPCODE

Major Name Space: **AJAX** (virtual clipboard name part 1)

Minor Name Space: **PRODUCT** (virtual clipboard name part 2)

Object Name (Name Part 3)	Description	Type
NUMBER	Current Product Number	Number – 7 digits
NAME	Current Product Name	String – max 40
PRICE	Current Product Price	Number – 11 digits, 2 decimals

Major Name Space: **ACCOUNTING** (virtual clipboard name part 1)

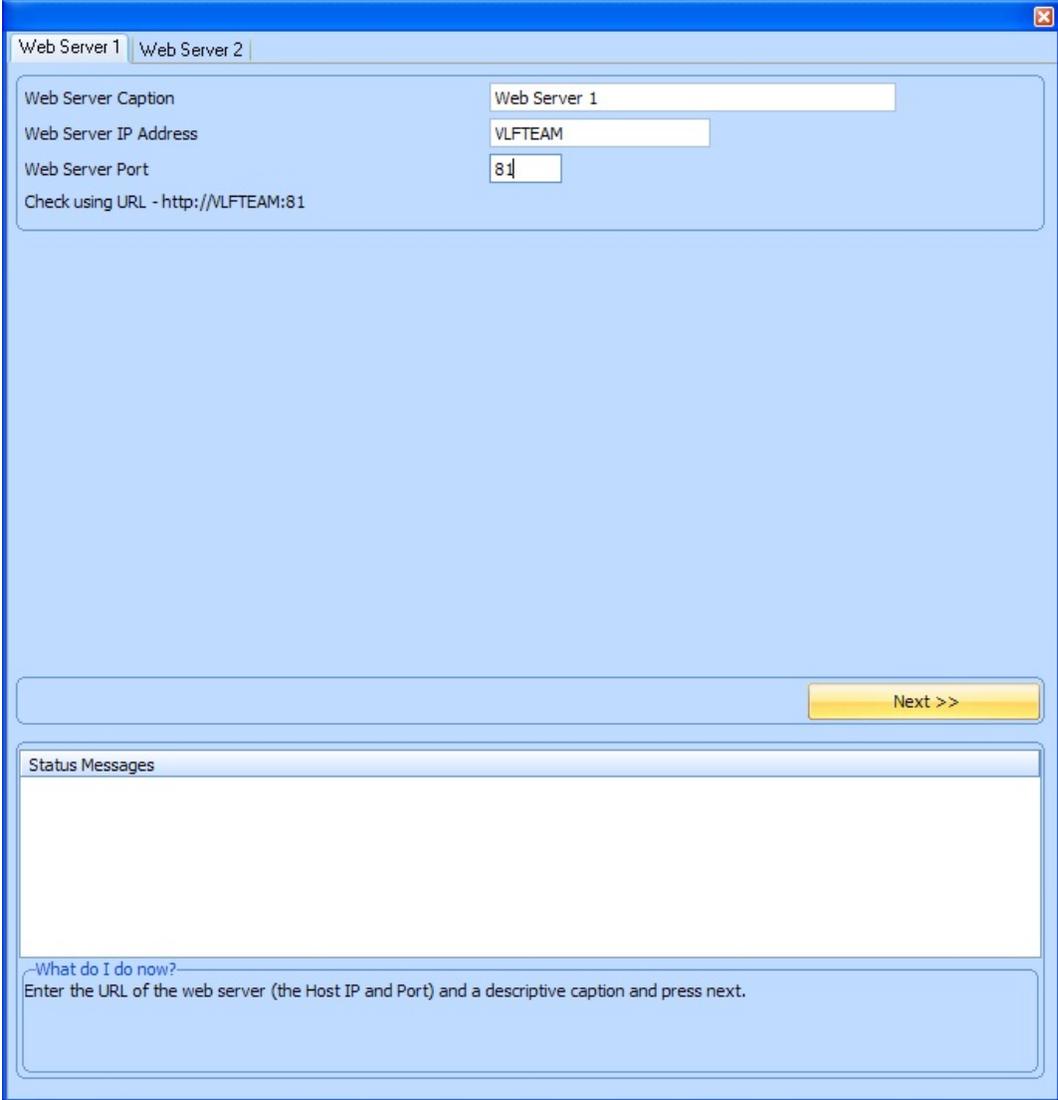
Minor Name Space: **USER** (virtual clipboard name part 2)

Object Name (Name Part 3)	Description	Type
USRPRF	System i User Profile	String – max 10
PRINTER	Default Printer	String – max 10

Web Configuration Assistant

The Web Configuration Assistant makes the web configuration of the Framework easier. Its use is optional, the alternative is to configure the VLF web using the Developer Preferences (in Framework Properties) and the VLF Administrator Console.

Initially, you specify the caption, IP address and port for the web server:



The screenshot shows a dialog box titled "Web Configuration Assistant" with two tabs: "Web Server 1" and "Web Server 2". The "Web Server 1" tab is active. It contains the following fields and text:

- Web Server Caption: Web Server 1
- Web Server IP Address: VLFTEAM
- Web Server Port: 81
- Check using URL - http://VLFTEAM:81

At the bottom right, there is a yellow button labeled "Next >>". Below the main configuration area is a "Status Messages" section, which is currently empty. At the bottom of the dialog, there is a "What do I do now?" section with the following text: "Enter the URL of the web server (the Host IP and Port) and a descriptive caption and press next."

The Web Configuration Assistant then detects information about the server and if necessary requests more details:

The screenshot shows a 'Web Configuration Assistant' window with two tabs: 'Web Server 1' and 'Web Server 2'. The 'Web Server 1' tab is active. The configuration fields are as follows:

- Web Server Caption: Web Server 1
- Web Server IP Address: VLFTEAM
- Web Server Port: 81
- Check using URL: -http://VLFTEAM:81/cgi-bin/lansaweb?procfun+VF_PR004+VFU0411+EX1+FUNCPARMS+VF_WKAct(A0010):'L'+V...
- The type of server: AS400
- The webserver's images URL path: /IMAGES
- The location of the images directory on the server: /LANSA_vifpgmlib/webserver/images/
- The webserver is on a different machine to this one. (checked)
- Do you want to use a mapped drive when uploading to the webserver? (unchecked)
- Private Working Folder: my project (with a red 'X' icon)
- Temporary Folder Name: vif_temporary_files

A yellow 'Next >>' button is located at the bottom right of the configuration section.

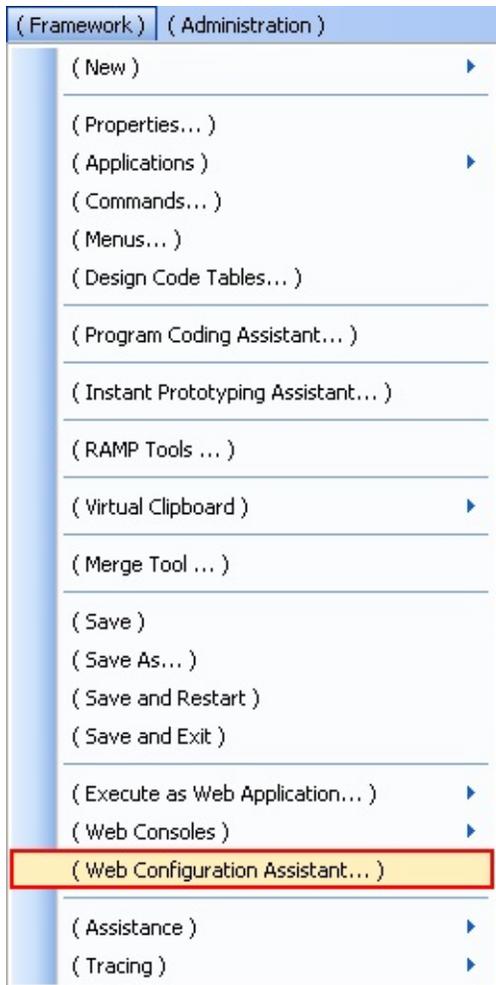
Status Messages

- ✓ A webserver has been found at this IP and Port
- ✓ LANSAs for the web is enabled for this lanssa installation
- ✓ LANSAs for the web is enabled for this lanssa partition and the VLF is installed
- ✓ The directory on the server associated with /Images was successfully detected.

What do I do now?
Enter a Private working folder name

The web server details specified using the Web Configuration Assistant are recorded in the Developer Preferences tab for the web server.

The Web Configuration Assistant can be accessed from either the (Framework) menu, or from the Framework Properties --> Framework Details tab.



Before you start, it will make configuration easier if you ensure that the following has been done:

- LANSA for the Web is installed, configured, and started on the web server.
- The partition you are working with on the web server is enabled for LANSA for the Web, and has been initialised.
- The partition you are working with on the web server is enabled for RDMLX.
- The Framework (EPC870 or later) has been imported into that partition on the web server.
- The web server is accessible from your PC.
- If the webserver is located on a different machine to your development machine, map a network drive to the web server, and record the drive letter used. See [Do you want to use a mapped drive when uploading to the webserver?](#)

The Web Configuration Assistant will define your web server(s) to the VLF. It

will create a VLF temporary files directory and your project directory within the images directory on the webserver.

Start by entering the web server's description, ip address and port, and press next. Then follow any subsequent instructions.

Warnings

- The Web Configuration Assistant cannot configure the VLF for multi-tier web environments.
- The Web Configuration Assistant may not be able to detect the location of the images directory, on the webserver. If so, you will need to supply this information.
- The detected values are saved during the last step. No changes will be saved unless you do the last step (Save my settings).
- The detector programs running on the web server need to have sufficient authority to create test files and subdirectories in the images directory.
- The first time a web server is accessed, there may be such a long delay that the Web Configuration Assistant thinks that it has failed. If you press the next button again, it may succeed on the 2nd attempt.

See:

[Web Server Caption](#)

[Web Server IP Address](#)

[Web Server Port](#)

[The type of server](#)

[The webserver's images path](#)

[The location of the images directory on the server](#)

[Do you want to use a mapped drive when uploading to the webserver?](#)

[Private Working Folder](#)

[Temporary Folder Name](#)

Web Application Start Options

Applies to **WAM** and **.NET**.

When you start a Web browser Framework application you can optionally add details to the URL that impact the way that your application executes.

Typically starting URLs are formatted like this example:

```
http://nnn.nn.nn.nn/...../vf_Sy001_System_ENG_BASE.htm
```

Optionally you can add parameters to the URL formatted like this:

```
http://nnn.nn.nn.nn/...../vf_Sy001_System_ENG_BASE.htm?  
Parm=value+Parm=value+ ..... +Parm=value
```

where the allowable parm and values are:

Parm	Comments
Partition=xxx	Specify the partition that the Framework is to connect to. VLF.WEB applications default this parameter to the partition that the Framework was saved from. VLF.NET applications default this parameter to DEM, so you should always specify this parameter.
Trace=XXXX	Specify the trace mode as Y or TRUE for a user level trace. Specify trace mode SYSTEM for a deeper system level trace. Note: In RAMP-TS sessions automatic error trapping is suppressed when TRACE mode has been turned on. This may allow script and HTML errors to be more easily isolated and debugged.
User=xxxx	Specify the user that the Framework is to use when connecting to the system. This value defines what user profile value prefills any log on dialog required by the Framework. If User= and Password= are both specified then the any log on dialog will be bypassed and the values passed directly to the

server. Security considerations may apply to the viability of specifying user and password details on URLs.

Developer=Y or
Developer=True

Indicates that the current user is a developer. Additional error handling and checking will be activated to aid the development process. Do NOT use this option in applications deployed to end-use or while performing performance testing.

Password=xxxx

Specify the password that the Framework is to use when connecting to the system. This value defines what password value prefills any log on dialog required by the Framework. If User= and Password= are both specified then the any log on dialog will be bypassed and the values passed directly to the server. If no password is required specify Password=none. Security considerations may apply to the viability of specifying user and password details on URLs.

SwitchTo=xxxx
SwitchObject=xxxxx
SwitchCommand=xxxxxxx

SwitchTo=, SwitchObject= and SwitchCommand= values may be specified to cause a Framework Switch instruction to be executed immediately that the Framework commences execution.

Switching to an instance level command associated with a business object is not possible because the user must manually select an instance first.

URLDATA1= xxxxx
through
URLDATA5=xxxxx

Specify up to five user data values that need to be passed into the application from the starting URL. The values specified in these parameters can be accessed by shipped User Imbedded Interface Poir (IIP) UFU0001 (or equivalent) as fields #URLDATA1 through #URLDATA5.

Executing WAM filters or command handlers can access these values by getting properties #Com_Owner.avURLData1 through

#Com_Owner.avURLData5.

These values may be specified in encoded URI format if required (eg: to handle imbedded blanks, etc).

Stats=Y	Specify the stats mode as Y to cause a basic statistics window to be displayed and updated while the application is executing. See following note 1 for details.
WAMHelp=Y	Activates end-user help for fields on WAMs.
TSUser=xxxxxxx NLUser=xxxxxxx	The overriding user profile that should be used in RAMP-TS or RAMP-NL applications when attempting to connect to the 5250 server. Note that RAMP-NL does not support the use of user profile containing the "@" (at) symbol. See Note 2 following for user profile default rules.
TSPassword=xxxxxxx NLPassword=xxxxxxx	The overriding password that should be used in RAMP-TS or RAMP-NL applications when attempting to connect to the 5250 server. See Note following for password default rules.
NLHostName=xxxxxxx	The symbolic host or connection name that newlook should use in RAMP applications when attempting to connect to the 5250 server. This symbolic name must be defined to newlook as a connection. The default is blank. If you use the NLHostName parameter you should not specify an NLIPAddress or NLPortNumber parameter value.
NLIPAddress	The IP address that newlook should use in RAMP applications when attempting to connect to the 5250 server. The default is the host name from which the Framework start page was started. If you use the NLIPAddress parameter you should not specify an NLHostName parameter value.
NLPortNumber	The IP port number that newlook should use in RAMP applications when attempting to connect to

the 5250 server. The default is 23. If you use an NLPortNumber parameter you should also specify an NLIPAddress value. If you use the NLPortNumber parameter you should not specify a NLHostName parameter value.

NLLiteClient

Indicates whether newlook should use a liteclient license when it is started, regardless of any default value determined when the Framework was generated.

Allowable positive values are NLLiteClient=Y or NLLiteClient=TRUE. Any other value will be taken as negative (ie: false).

NLIniFile

Specify Name of the newlook .INI file to use with your RAMP application. This value will override the one specified in the Server definition.

NLUpdateFile

Specify Name of the nlupdate.txt file to use with your RAMP application. This value will override the one specified in the Server definition.

NLCodeBase

Specify Name of the newlook.cab file to use with your RAMP application. This value will override the one specified in the Server definition.

DEBUG=Y,<<host IP>>:
<<port>>

Allows WAM command handlers and filters to be debugged remotely.

The <<host IP>> is the ip of the computer where the Visual LANSAs development environment (IDE) is running.

On LANSAs systems earlier than SP5 + EPC830, the <<port>> is the port used by the listener (see the Communications Administrator for the port used by your listener).

An example of using remote debug for WAMS on your own PC would be to add:

```
+DEBUG=Y,127.0.0.1:4545 to the URL
```

On LANSAs systems where EPC830 has been applied and later systems, LANSAs has a separate service for debugging

The <<port>> for this service can be found in the IDE, under Options, Settings, Debug. It is often port 51234.

An example of using remote debugging for WAMS on your own PC would be to add:

```
+DEBUG=Y,127.0.0.1:51234 to the URL
```

Theme=XXXX

Used in .NET applications only. Ignored in other applications.

Specifies the color and style theme to be used in the executing application.

If not specified, the theme specified in the Framework definition by the application designer is used.

Allowable values are 2007BLUE, 2007SILVER, 2007GRAPHITE and 2007OLIVE.

NETURL

When a VLF.NET application is started, the NETURL=value indicates the place from which VLF related HTML pages and JavaScript files should be loaded.

For a VLF developer this is most typically their private folder, for example

```
NetURL=http://MYHOST/images/MyPrivateFolder
```

In a production application this is most typically the LANSAs for web images folder, for example

```
http://MYSERVER/Images/
```

TSPrivateSet=

Specifies the [Private Definition/aXes Project Folder](#) that any RAMP-TS session started should use.

When used, specify just the private definition/aXes folder name, not the complete path name. The path

to the folder is implicit. If not specified, the private definition/aXes folder used will default to the value associated with the RAMP-TS server flagged as the “deployment server” at the time the framework was last saved.

TSPrivateShared=

This property corresponds to the [Contains SHARE Object](#) option in a RAMP-TS server definition which allows you to have the SHARED object in your own private definition/aXes folder. Allowable values are True and False.

If you use TSPRIVATESHARED=TRUE and the file is not found you will get an error.

TSLoadPath=

Specifies the load path that any RAMP-TS session started should use. If not specified, the load path used will default to the value associated with the RAMP-TS server flagged as the “deployment server” at the time the framework was last saved.

TSIPAddress=

Specifies the IP address that any RAMP-TS session started should use. If not specified, the IP address will default to IP address of the VLF-WEB session. Typically TSIPADDRESS and TSPORTNUMBER are used together. The VLF-WEB session and the RAMP-TS session must be in the same domain. The RAMP-TS server’s IP address needs to be defined as a trusted server.

TSPortNumber=

Specifies the port number any RAMP-TS session started should use. If not specified, the port number used will default to the value associated with the RAMP-TS server flagged as the “deployment server” at the time the framework was last saved. Typically TSIPADDRESS and TSPORTNUMBER are used together.

TSUSEHTTP=

TSUSEHTTPS=Y is for RAMP-TS only. It indicates that aXes should be loaded using https:\\ as

	<p>the protocol. If used, the VLF-WEB (LANSA for the Web) application must also use https:\\.</p> <p>Additionally both aXes and LANSA for the web must load from the same domain (that is literally from the exactly the same name on the URL).</p>
TOUCH=Y	<p>Enables and disables Framework functionality to make it more touch friendly. For an overview please refer to Touch Device Considerations.</p> <p>If the TOUCH parameter is not present in the URL the VLF will guess whether it's running on such device.</p>
ZOOM=	<p>The CSS zoom to apply to the VLF. The default zoom value is 1.4.</p>
CHKPSWEXPIRY = Y CHKPSWEXPIRY = Yes	<p>Set to Y to compare the expiry date of the password typed in during WEB sign on with the current date. If the difference is less than the value specified in ISSUEWARNING, a warning message is issued.</p> <p>Refer to the shipped Web IIP for User signon function UFU0001 in process UF_SYSBR.</p>
ISSUEWARNING = nnn	<p>Specifies how many days before the password expiry date a message will be issued during Web signon to warn that the password is about to expire</p> <p>Refer to the shipped Web IIP for User signon function UFU0001 in process UF_SYSBR.</p>
ALLOWPSWCHG = Y ALLOWPSWCHG = Yes	<p>When this parameter is set to Yes, the end-user can change the IBM i password. The signon dialog will contain fields in which to enter the old and the new password, and a Change button.</p> <p>Refer to the shipped Change Password IIP function UFU0006 in process UF_SYSBR.</p>
IBMISERVER	<p>Name or IP address of the IBM i Server Mapper for</p>

password expiry and password change.

IBMIPORT

Port of the server specified in IBMISERVER.

Also see [The URL to start my deployed VLF web browser application is too complex for users to reliably type in to their browsers.](#)

Note 1: Using Stats=Y

Using Stats=Y causes a basic statistics window to be displayed and updated while the application is executing.

The details shown reflect elapsed times as measured by the client system.

SERVER-SIDE: This is the time between when the web form was submitted to the server and when it arrived back at the client system, therefore it includes the server side processing time and all send and receive communication delays.

CLIENT-SIDE: This is the time between when the web form arrived at the client and when all client side processing of it was completed (excluding any additional time taken by the browser to draw the web form).

OVERALL: This is the time between when the web form was submitted to the server when all client side processing of it was completed (excluding any additional time taken by the browser to draw the web form). OVERALL is approximately equal to SERVER-SIDE + CLIENT-SIDE.

In very broad terms these statistics can be used to identify problem areas in your application such as:

High SERVER-SIDE values may indicate that:

- Your filter or command handler may be too complex and taking too long to execute on the server.
- The communications delay between your client system and your web server is too long.

High CLIENT-SIDE values may indicate that:

- Your filter or command handler is causing the client side to do too much processing. A cause of this may be the overloading of the client side by browse lists with too many entries in them or by client systems with slow or busy CPUs.

Note 2: RAMP-TS and RAMP-NL User Profile and Password

defaulting

The user profile and password used by RAMP-TS or RAMP-NL when attempting to connect to a 5250 server are defaulted as follows:

- If your Framework is using Framework security and the user needs to log on, the user profile and password they use to log on become the initial default values.
- For RAMP-NL only, the user profile that the user logs on to the framework as may have a specific RAMP-NL user profile or password associated with it. These values, if present, will override the initial defaults.
- Any user (TSUSER/NLUSER) or password (NLPASSWORD/TSPASSWORD) parameters specified on the start up URL will override any of the preceding default values.
- Finally, a user profile and password may be returned by the system logon validation program. These values will completely override any other default or deduced user or password values.

Tip: If you are having problems with RAMP user profiles or passwords then try executing your application with TRACE=Y to see the values that are being used.

Program Coding Assistant

The Program Coding Assistant is designed to help you more easily create filters, command handlers and snap in instance lists by generating code for:

- Complete Visual LANSA components.
- Code fragments for operations you commonly perform.
- Complete LANSA for the Web Application Modules (WAMs).

Using the Program Coding Assistant to generate code involves these steps:

[Step 1. Select Object](#)

[Step 2. Select the Target Platform](#)

[Step 3. Select Type of Code – Review Abstracts](#)

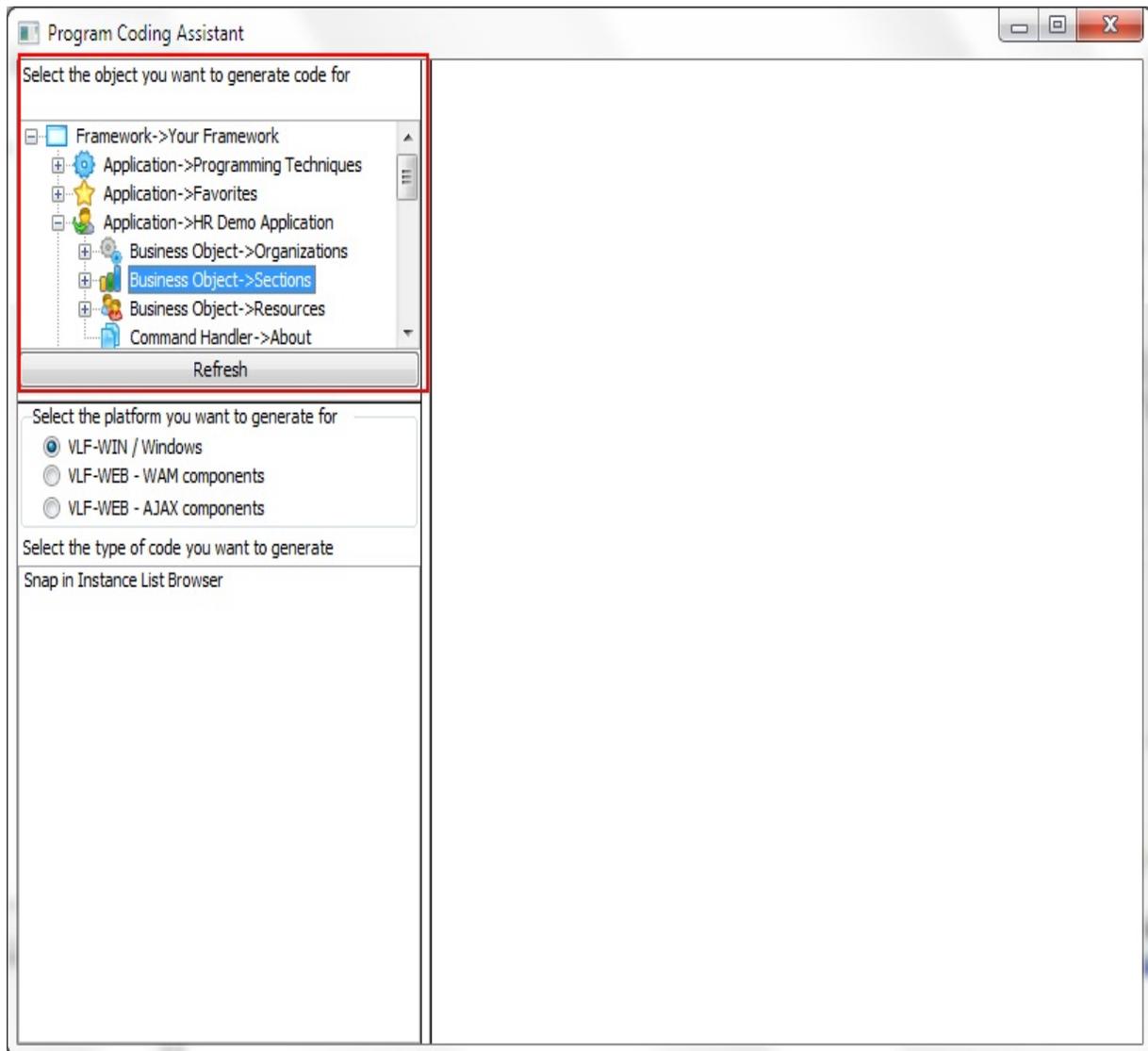
[Step 4. Fill in Prompts](#)

[Step 5. Generate Code](#)

[Step 6. Create the Component](#)

Step 1. Select Object

Select the application, business object, filter or command handler that you want to generate code for:



Step 2. Select the Target Platform

Select the platform that you want to generate code for:

What? This assistant produces the code for a basic business object command handler. Depending upon the options you use it may or may not require you to manually add code to it later.

Business object command handlers execute a *command* against one or more selected *business object instances* (e.g. *Display an Employee, show the History of a Product, Print all selected Orders*).

In a many Visual LANSA framework command handlers are presented to the user like this (in the area circled in red):

Number	Name
41404	Gilvan Black
42364	Freddy Brown
40290	Fred Bloggs
40370	Veronica Brown
40321	John Blake

The essential business object Details (41404-Gilvan Black)

Employee Number: 41404
Employee Surname: Black
Employee Given Name(s): Gilvan
Street No and Name: 22 Nelson Street
Suburb or Town: Mandeville
State and Country: NSW

Step 3. Select Type of Code – Review Abstracts

Select the type of code you want to generate:

The screenshot shows the 'Program Coding Assistant' window. On the left, a tree view shows the selection path: Business Object->Sections->Command Handler->New. Below this, radio buttons allow selecting the platform: VLF-WIN / Windows (selected), VLF-WEB - WAM components, and VLF-WEB - AJAX components. A red box highlights the 'Select the type of code you want to generate' section, where 'Basic Command Handler' is selected. On the right, the 'Basic Command Handler' abstract is displayed, explaining that it produces code for a basic business object command handler. Below the abstract, a smaller screenshot shows a sample application window titled 'The essential business object' with a search form and a table of employee data.

What? This assistant produces the code for a basic business object command handler. Depending upon the options you use it may or may not require you to manually add code to it later.

Business object command handlers execute a *command* against one or more selected *business object instances* (e.g. *Display an Employee, show the History of a Product, Print all selected Orders*).

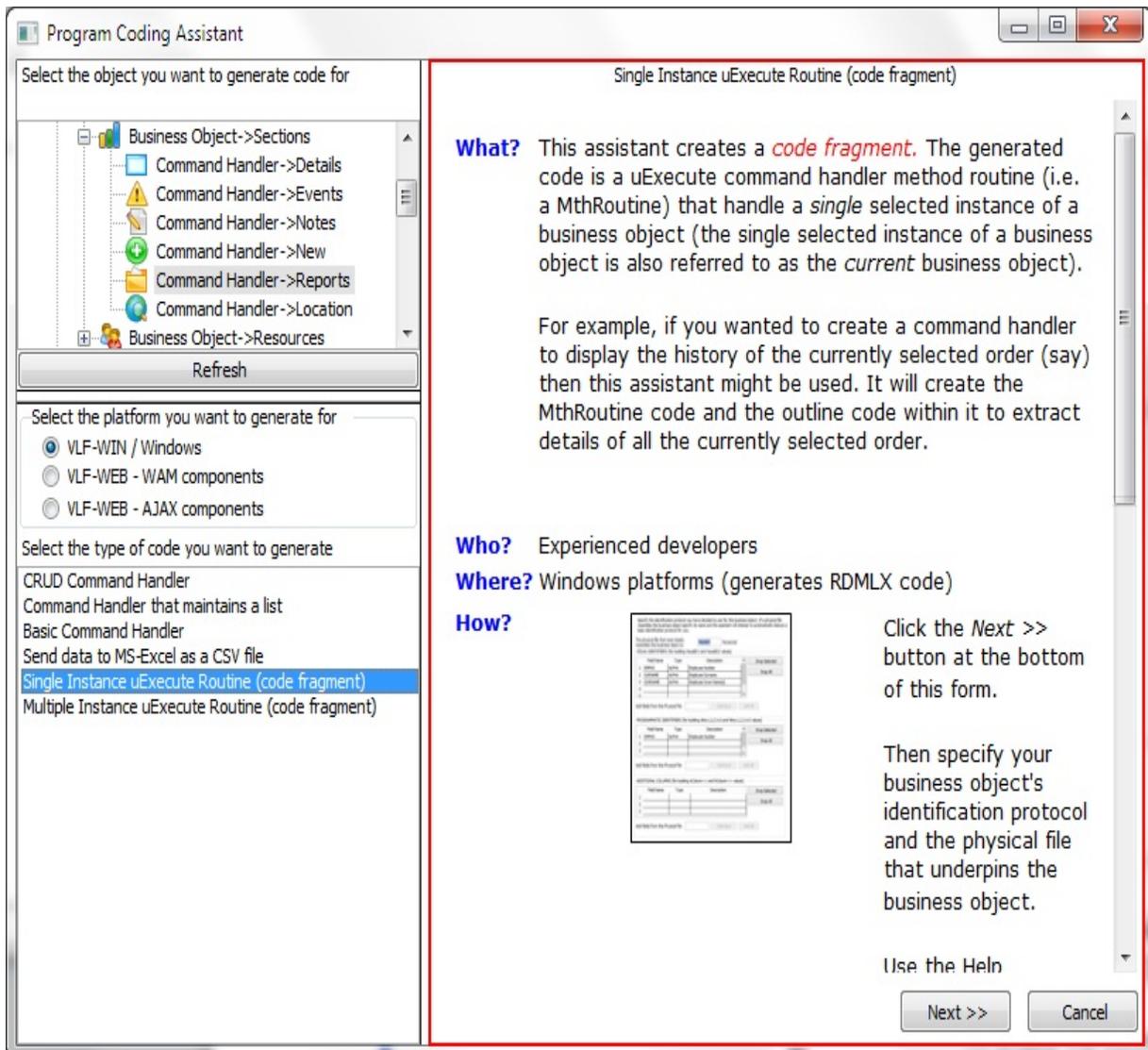
In a many Visual LANSa framework command handlers are presented to the user like this (in the area circled in red):

Number	Name
41404	Gillian Black
42304	Freddy Brown
40200	Fred Bloggs
40370	Veronica Brown
40321	John Blake

The essential business object Details (41404-Gillian Black)

Employee Number: 41404
Employee Surname: Black
Employee Given Name(s): Gillian
Street No and Name: 22 Milton Street
Suburb or Town: Mandeville
State and Country: NSW

To get more details about the type of code click on it and review the abstract that appears on the right hand side of the window:



Single Instance uExecute Routine (code fragment)

What? This assistant creates a *code fragment*. The generated code is a uExecute command handler method routine (i.e. a MthRoutine) that handle a *single* selected instance of a business object (the single selected instance of a business object is also referred to as the *current* business object).

For example, if you wanted to create a command handler to display the history of the currently selected order (say) then this assistant might be used. It will create the MthRoutine code and the outline code within it to extract details of all the currently selected order.

Who? Experienced developers

Where? Windows platforms (generates RDMLX code)

How?



Click the *Next >>* button at the bottom of this form.

Then specify your business object's identification protocol and the physical file that underpins the business object.

Use the *Help*

Next >>

Cancel

Step 4. Fill in Prompts

Fill in the prompt(s) appropriate for type code that you want to generate:

The screenshot shows the 'Program Coding Assistant' dialog box. The left pane shows a tree view of business objects, with 'Business Object->Resources' selected. The right pane is titled 'CRUD Command Handler' and contains several sections for configuration. A red box highlights the 'VISUAL IDENTIFIERS' section, which includes a text prompt, a table with 3 rows, and buttons for 'Drop Selected' and 'Drop All'. Below this are sections for 'PROGRAMMATIC IDENTIFIERS' and 'ADDITIONAL COLUMNS', each with a similar table and buttons. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

Select the object you want to generate code for

- Business Object->Sections
- Command Handler->Details
- Command Handler->Events
- Command Handler->Notes
- Command Handler->New
- Command Handler->Reports
- Command Handler->Location
- Business Object->Resources

Refresh

Select the platform you want to generate for

- VLF-WIN / Windows
- VLF-WEB - WAM components
- VLF-WEB - AJAX components

Select the type of code you want to generate

- CRUD Command Handler
- Command Handler that maintains a list
- Basic Command Handler
- Send data to MS-Excel as a CSV file
- Single Instance uExecute Routine (code fragment)
- Multiple Instance uExecute Routine (code fragment)

CRUD Command Handler

Specify the identification protocol you have decided to use for this business object. If a physical file resembles this business object specify its name and the assistant will attempt to automatically deduce basic identification protocol for you.

The physical file that most closely resembles this business object is:

VISUAL IDENTIFIERS (for building VisualID1 and VisualID2 values)

	Field Name	Type	Description
1			
2			
3			

Drop Selected

Drop All

Add fields from this Physical File Add Keys Add All

PROGRAMMATIC IDENTIFIERS (for building AKey1,2,3,4,5 and NKey1,2,3,4,5 values)

	Field Name	Type	Description
1			
2			
3			

Drop Selected

Drop All

Add fields from this Physical File Add Keys Add All

ADDITIONAL COLUMNS (for building AColumn<> and NColumn<> values)

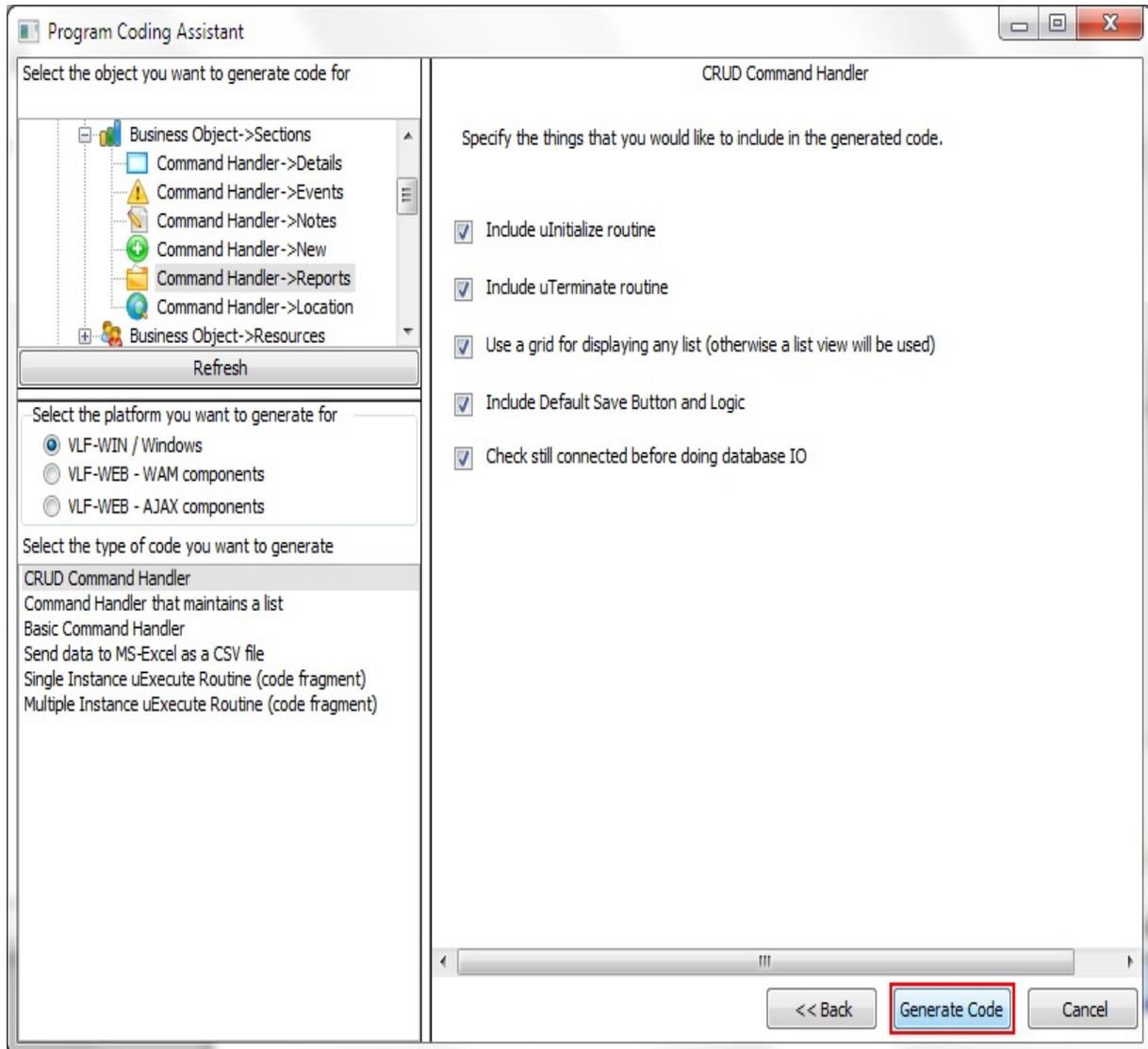
	Field Name	Type	Description
1			

Drop Selected

<< Back Next >> Cancel

Step 5. Generate Code

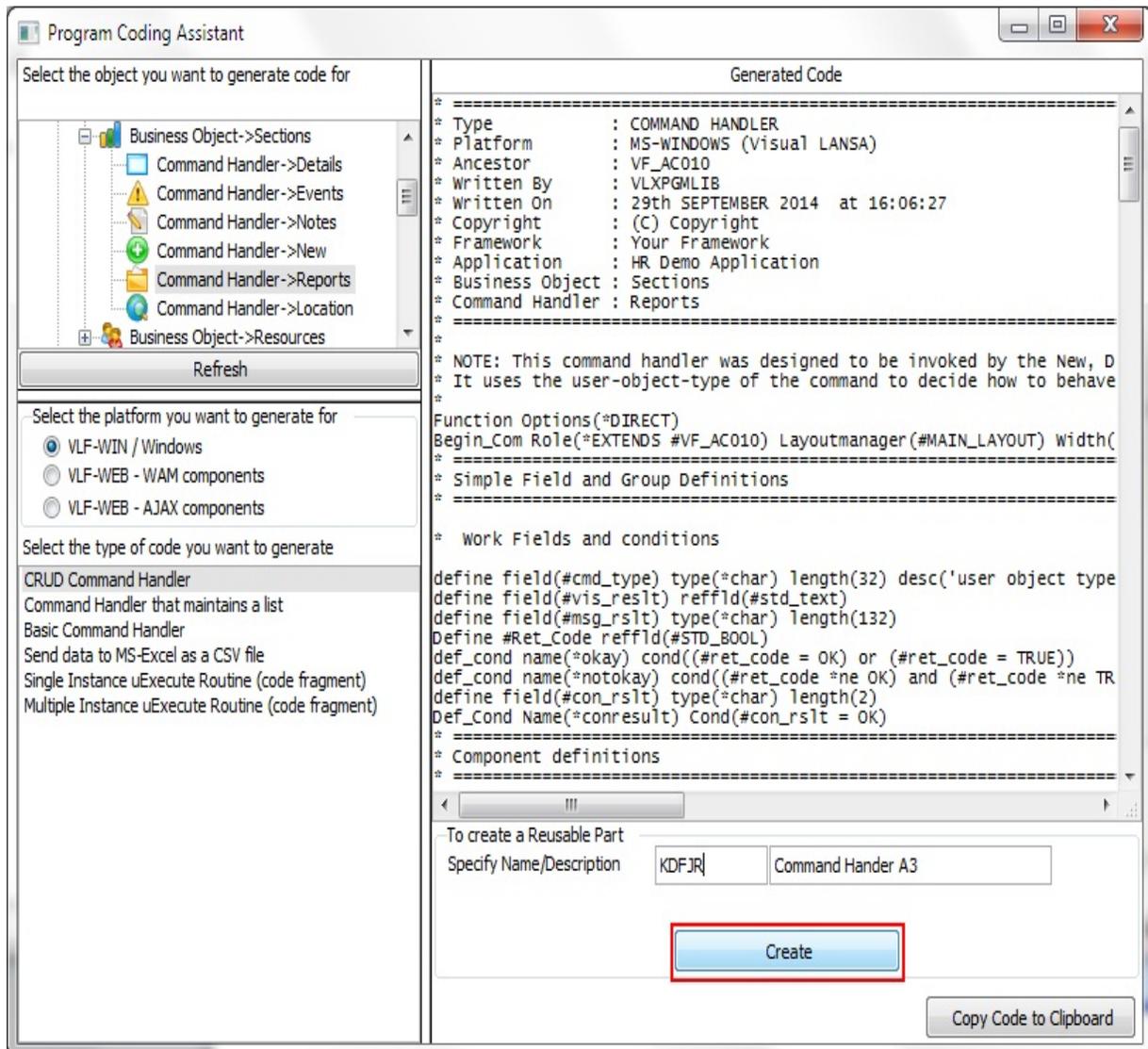
Generate the code:



Step 6. Create the Component

Create the Visual LANSA component to contain the generated code.

Alternatively you can paste the generated code to the clipboard and then to the Visual LANSA editor:



Code Tables

Most commercial applications involve the use of codes and their decodes in many situations. For example:

Country Code	Country Decode
1	North America
61	Australia
44	Great Britain
31	Netherlands
64	New Zealand
65	Singapore
81	Japan

Currency Code	Country Decode
USD	US Dollar
GBP	Great Britain Pound
AUD	Australia Dollar
JPY	Japanese Yen

Sex Code	Sex Decode
M	Male
F	Female
U	Unspecified

--	--

Document Code	Document Decode
DOC	MS-Word Document
PPT	MS-PowerPoint Document
RTF	Rich Text Document
TXT	Text Document

Application end-users need to be able to select a code from a displayed list of decodes, validate it using a referential integrity check and decode it to present the decode on a form or in a report

Typically codes come in relatively short lists of less than 100 items and thus end-users can select the one that they want from for example a group or radio buttons or a drop-down combo box.

However some code tables are large. For example a customer code table could store 10,000 customers. To handle large code tables you need to provide the end-user with intelligent prompting capability that allows them to quickly locate the customer they are interested in.

- [Framework Code Tables](#)
- [Code Table Data Flow](#)
- [Setting up a Code Table](#)
- [Using a Code Table in your application](#)
- [Advanced options when setting up a code table](#)
- [Frequently asked Questions about Code Tables](#)
- [Using Assistants to handle more complex "codes"](#)

Framework Code Tables

The Framework Code Tables provide a number of facilities to make it possible to more productively handle codes, decodes and prompting in your application.

Code Tables are:

Fully configurable

Designers simply specify the keys and data items that define the code table to the Framework.

To design a Currencies code table the Framework designer needs to define the Currency Code as a key and the Currency Description as a Data item.

May be sourced from your database tables

The data inside code tables can come from your existing database tables or the Framework can store it automatically.

The Currencies code table could be stored and managed by the Framework itself in its own database table or sourced from one of your own database tables (or from anywhere else that your computer can access).

The format of the code table you define does not have to be the format of the database table it is sourced from. Sometimes the data in a table is sourced from "hard coded" program values. For example the Sex table mentioned previously may be defined as data within a program only and never be actually stored anywhere.

Can be updated in deployed applications by end-users

Optionally end-user administrators can be allowed to update information stored in code tables.

The user interface for updating the Code Table is provided by the Framework. The final updating and storage of the information in the table can be done automatically by the Framework or by a program that you provide.

Can be visualized by you to your end-users in various ways

Codes can be visualized to end-users in different ways, ranging from a group of radio buttons (eg: the Sex code table) to a drop-down combo box (eg: the Currencies code table).

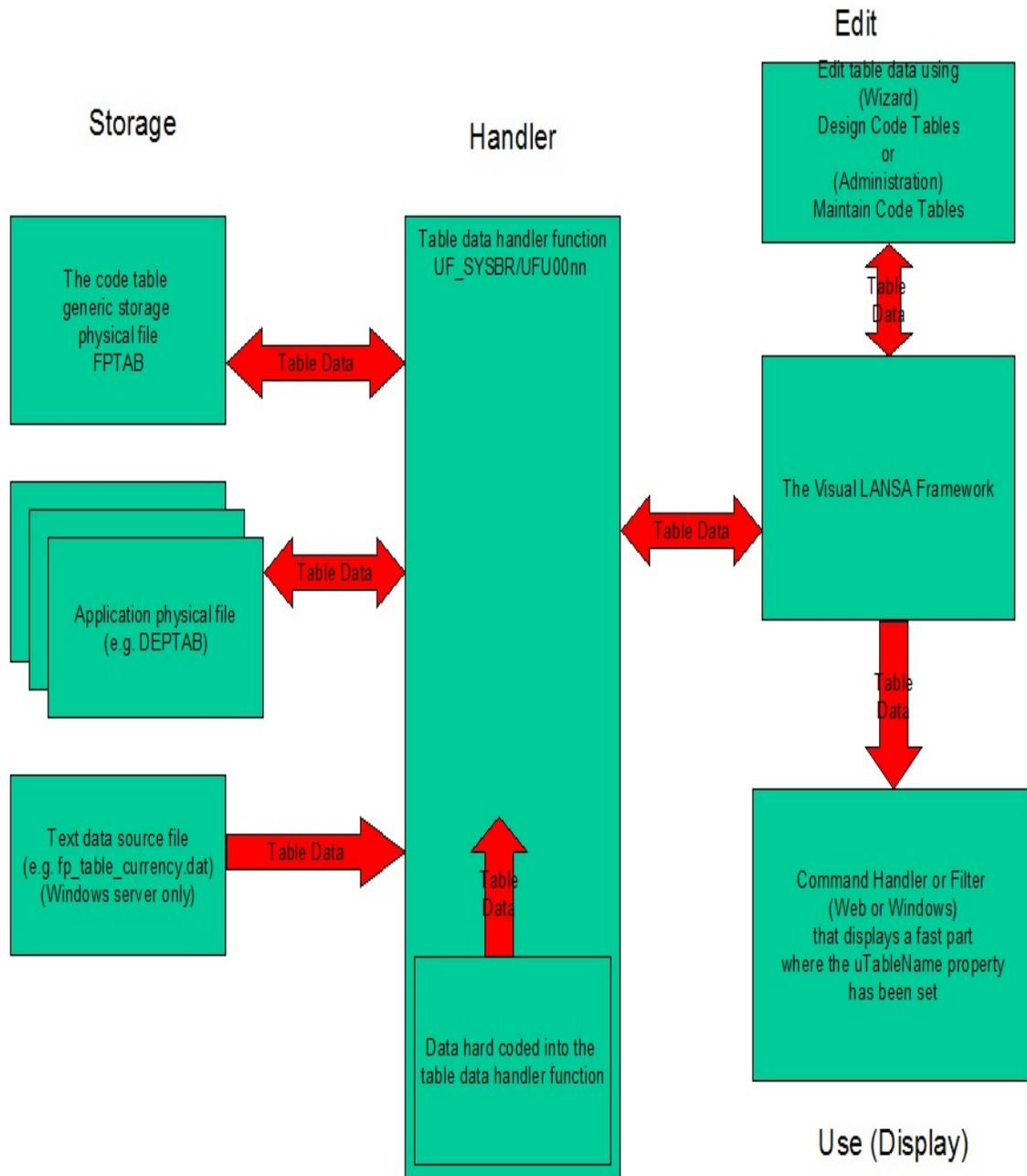
Are an alternative to defining simple small database tables

Code tables are a way of defining simple tables quickly and easily. They are suited to tables with less than 100 data items, with small numbers of fields (2 - 6), that do not require validation checks when the table data is entered, virtual

fields, or logical views.

If you require functionality beyond these limits, you should define your database table as LANSAs, and write your own prompt assistant program.

Code Table Data Flow



Setting up a Code Table

To define a code table

- Start your Framework as a designer.
- Select the (Framework) and then (Design Code Tables) menu options.

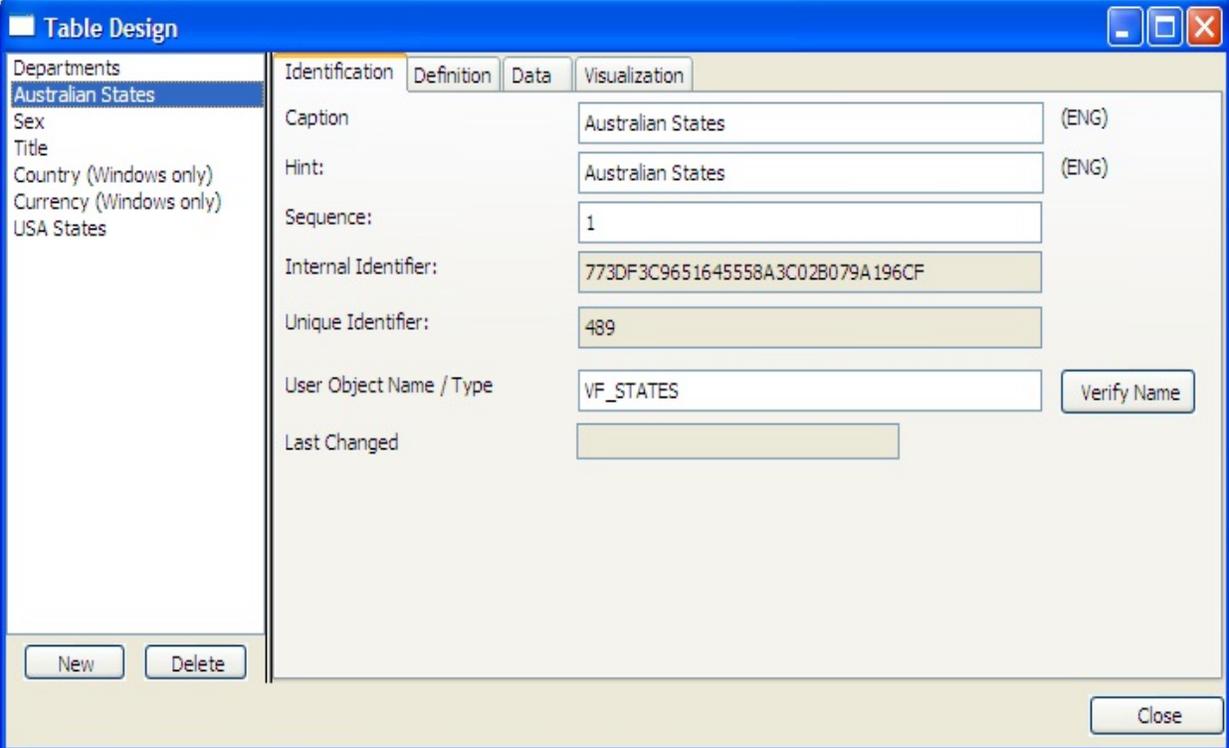
The resulting form allows you to define new code tables and change existing ones:

- [Step 1. Create a Table](#)
- [Step 2. Define a Table](#)
- [Step 3. Enter data into a code table](#)

Step 1. Create a Table

Use the new button to create a new table. On the identification tab give your table a caption and most importantly a User Object Name/Type. The example shown is the shipped Australian States code table.

The User Object Name/Type will be used by programs to identify which table they want to use.



The screenshot shows the 'Table Design' window with the 'Identification' tab selected. The table being configured is 'Australian States'. The 'User Object Name / Type' field is set to 'VF_STATES'. The 'Caption' and 'Hint' fields are both set to 'Australian States'. The 'Sequence' is set to '1'. The 'Internal Identifier' is '773DF3C9651645558A3C02B079A196CF' and the 'Unique Identifier' is '489'. The 'Last Changed' field is empty. There are 'New', 'Delete', and 'Close' buttons at the bottom of the window.

Field	Value	Language
Caption	Australian States	(ENG)
Hint	Australian States	(ENG)
Sequence	1	
Internal Identifier	773DF3C9651645558A3C02B079A196CF	
Unique Identifier	489	
User Object Name / Type	VF_STATES	
Last Changed		

Step 2. Define a Table

Define the field structure of this table, and some other details:

- Go to the Definition tab. For this example the field name can be anything, as long as it is unique within this table. This is because this table will use the default table data storage function (UFU0010). For tables whose data is stored on an application database file (e.g. table VF_DEPTAB) it is easier if the field names of the code table match those of the application's database file.
- Mark as keys those fields that will uniquely identify each row of data in the table. There must be at least one key field. Key fields must not be greater than 32 characters or 15,5 numeric. There must be no more than 5 key fields.
- Specify the field lengths and decimals if necessary.
- Leave the language field as "No Language Field – Monolingual Table Data"
- Uncheck Read_Only
- Leave the Function handling table data storage as the default – UFU0010
- Press the Save button

You have now defined a code table.

Table Design

Departments
Identification Definition Data Visualization

Australian States
 Sex
 Title
 Country (Windows only)
 Currency (Windows only)
 USA States

Save
Delete
Move Up
Move Down

Field Name	Key	Caption	Type	Max Length	Decimals	Upper Case Only
CODE	<input checked="" type="checkbox"/>	State Code	Alpha	3		<input checked="" type="checkbox"/>
DESCRIPTN	<input type="checkbox"/>	Description	Alpha	40		<input type="checkbox"/>
MYSEQ	<input type="checkbox"/>	Sequence	Numeric	3		<input type="checkbox"/>
	<input type="checkbox"/>		Alpha			<input type="checkbox"/>
	<input type="checkbox"/>		Alpha			<input type="checkbox"/>
	<input type="checkbox"/>		Alpha			<input type="checkbox"/>

Inactive Table Entry Indicator: No field indicates Inactive Table Entry Read C

Language Field: No Language Field - Monolingual Table Data

Function handling table data storage:

- PSLUTL - Personnel System Utility Programs
- UF_SYSBR - Imbedded Interface Points
 - UFU0001 - Shipped Web IIP for User Sign
 - UFU0003 - MTXT Enroller
 - UFU0010 - Default table data handler
 - UFU0011 - Deptab table data handler

New
Delete
Refresh

Close

Using a Code Table in your application

Once the table and its data content are defined to the Framework you can start to use it within your applications in various ways:

- [As a part of a referential integrity check](#)
- [As a decode or lookup operation in your programs](#)

As a part of a referential integrity check

Code tables may be used to define standard referential integrity checks in the LANSAs repository or in LANSAs RDML functions. For example you might need to check that a state code "NT" that has arrived from someone else's application is a valid state code.

Only tables with data that is stored in a physical file (either user defined or the generic Framework table storage file FPTAB) allow referential integrity checks.

In the RDML, a check against a code table named VF_STATES would look like:

```
*Use the logical view (kya) keyed by:  
*Table name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ..., field name  
CHECK_FOR in_file(FPTABkya) with_key('VF_STATES' 'NT') IF_STATUS  
...  
ENDIF
```

A LANSAs file validation check against the employee table would look like this

New

PSLMST

Description: Check employee number is valid

Sequence: 2

Rule definition | Actions

Validation usage

When inserting: Always apply rule (ADD)

When updating: Always apply rule (CHG)

When deleting: Never apply rule

Define rules

File name: FPTABNMA

PSLMST Field or literal	FPTABNMA Key field
EMPNO	FP_ETABL - Table
	FP_EPTNAM - Property Name
	FP_EKEY1 - Key Value 1
	FP_EKEYN1 - Numeric Key 1
	FP_EKEY2 - Key Value 2
	FP_EKEYN2 - Numeric Key 2
	FP_EKEY3 - Key Value 3
	FP_EKEYN3 - Numeric Key 3
	FP_EKEY4 - Key Value 4
	FP_EKEYN4 - Numeric Key 4
	FP_EKEY5 - Key Value 5
	FP_EKEYN5 - Numeric Key 5

As a decode or lookup operation in your programs

Code tables may be accessed by programs operating in non-visual contexts. For example a batch reporting programming may need to convert the currency code "USD" to the description "US Dollars" before printing it on a report.

Only tables with data that is stored in a physical file are accessible by such functions.

*Use the logical view (nma) keyed by:

*Table name, field name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ...

```
FETCH fields(#FP_EPTVAL) from_file(FPTABnma) with_key('VF_CURRE
```

```
Change #MyDescriptionField #FP_EPTVAL
```

For more examples, refer to [Frequently asked Questions about Code Tables](#)

Advanced options when setting up a code table

[Multilingual Data](#)

[Sequencing](#)

[Choosing the description field](#)

[Creating your own table data handler function](#)

[Read Only Tables](#)

[Inactive Table Entry indicator](#)

[Creating Your Own Table Data Handler Reusable Part](#)

Multilingual Data

In multilingual applications it may be necessary to have multilingual table data. In such situations it is common for a single code to have several descriptions, or sometimes, a different set of codes for different languages. The code tables system handles this by having one record for each language for each code.

Consider the table data for the VF_TITLE table

Title Code Language Title Description

MR	ENG	Mr.
MR	DEU	Herr.
MRS	ENG	Mrs.
MRS	DEU	Frau.
MISS	ENG	Miss.
MISS	DEU	Frl.

The language field parameter (on the table definition tab) is used to designate which of the key fields is a language code field. (the field called "language" in this case).

At run time only the table entries with the correct language code for the user will be displayed.

If multilingual table data is not required, set the language field to the value "No Language Field - Monolingual Table Data". In this case, all table entries will appear at run time.

A table doesn't necessarily have to be multilingual just because the partition is multilingual.

Sequencing

In the code table visualization tab you can specify which of the table fields should be used to sequence the table entries when they are displayed to the end-user at run time.

Choosing the description field

In the code table visualization tab you can specify which of the table fields should be used as the caption for the table entries when they are displayed to the end-user at run time.

Creating your own table data handler function

Your table can share the default table data handler or you can create your own function. The simplest method of creating a table data handler is to copy one of the following example functions

Tables where the data is stored in the shipped generic table data file (FPTAB) can share the default table data handler UF_SYSBR/UFU0010, or use their own version of it.

Tables where the data is stored in an application's physical file can create their own table data handler function. See the data handler function UF_SYSBR/UFU0011 for an example. This function reads and writes data from/to file DEPTAB, for the table DEPTAB.

Tables where the data comes from hard coding in the table data handler function can create their own table data handler function. See the data handler UF_SYSBR/UFU0012 for an example of this type of data handler. It supplies the data for the SEX Table.

Tables can also get their data from a flat file, using a table data handler function similar to UF_SYSBR/UFU0013, provided the end-user is running the Windows Framework and has access to the flat file.

Table data handler functions may or may not allow data to be updated. UFU0012 and UFU0013 are examples of table data handler functions that supply table data but will not store edited table data.

Read Only Tables

This table property is set in the table definition tab.

It is sensible to make a table read-only if its table data handler has no method for saving or updating data. The read only option could also be used for those tables where the complete data set has been added and saved, and the designer does not wish to allow the administrator to modify the table data in future.

Inactive Table Entry indicator

In some circumstances it is useful to be able to distinguish between table entries that are live, and inactive table entries that are present only because they belong to historical data. The inactive table entries need to be present so that historical data is displayed correctly, but they shouldn't be used when creating new data.

If you want to flag some of the table entries as inactive, create a field to hold this information (a one character field) and then choose that field in the Inactive Table Entry indicator drop down.

Creating Your Own Table Data Handler Reusable Part

A reusable part code table data handler can source data from any source that a data handler function can, and if it is enabled for RDMLX, it can process Unicode table fields.

A Framework code table can use the default reusable part data handler, UF_TDH01, or use a custom data handler. If UF_TDH01 is used as a table's data handler, data will be written and read from the shipped VLF Code Table files FPTAB (main data file) and FPTABU (Unicode value file).

If a custom data handler reusable part is created, it must extend VF_AC024, the Table Data Handler ancestor class, and redefine the methods included in the ancestor class. The default behaviour can be optionally included by calling the ancestor methods in the redefined methods first. See comments in UF_TDH01 for more details.

See [Code Table Definition/ Reusable Part Data Handler \(ID\)](#) and [Code Table Definition/ Use a Reusable Part](#).

Frequently asked Questions about Code Tables

Q: Are code table real database files?

A: No, code tables are abstract or conceptual definitions only. You define to the Framework the columns that are in the code table and indicate which ones define the unique key for a row in the code table. The table data is frequently stored in a real database file, though it may share the database file with other tables.

Q: Where does the data in a code table come from?

A: It can come from anywhere. By default the Framework is shipped with an RDML function that will store code table data inside a single database table. However you can supply your own data storage function that can source the code table data from anywhere that you want. This type of function is referred to as a Table Data Handler Function.

Q: How do I create a Data Storage function?

A: A Table Data Handler function is a normal LANSAs RDML function that communicates with the Framework using a pre-defined protocol. If you want to create your own Table Data Handler function, see process UF_SYSTR functions UFU0010 – UFU0015 for examples.

Q: Can Data Storage Functions interact with the end-user?

A: No. Data storage functions are designed to act as data retrieval and update routines that can work in many different contexts. For example they can be invoked as a remote procedure by a Windows based Framework application or on a remote server as part of browser based application. This means that they need to be able to operate in contexts where no user interface is available to them.

Q: What are the benefits in using code tables?

A: The main benefits in using the Framework code table system are simply in improved productivity and consistency. By using a standard shipped architecture for code table maintenance you can develop and maintain applications more rapidly and avoid the cost and complexity of developing your own code table system.

Q: How do I interface an external LANSAs function (or LANSAs for the web function) with the Framework generic table data file (FPTAB)?

A: The data in FPTAB is unusual in that it contains one record for every non-

key cell in the table.

For example, in the Australian States table there are the fields

```
CODE (a key)
DESCRIPTN
MYSEQ
```

Each state will be represented as two records in FPTAB: One record for DESCRIPTN and one record for MYSEQ. Both records will contain all the key data (CODE in this case).

Akey1 (FP_EKEY1)	Nkey1(FP_EKEYN1)	Other Property keys2 Name(FP_EPTNAM) - 5	Alpha property v (FP_EPTV
NSW		DESCRIPTN	New South Wales
NSW		MYSEQ	
QLD		DESCRIPTN	Queenslan
QLD		MYSEQ	
...			

ExamplesI want to check that a currency code entered by a user is valid

*Use the logical view (kya) keyed by:

*Table name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ..., field name

```
CHECK_FOR in_file(FPTABkya) with_key('VF_CURRENCY' #MyCurrency
IF_STATUS *EQUALKEY
```

```
ENDIF
```

I want to display the description of a currency code that I have read from somewhere

*Use the logical view (nma) keyed by:

*Table name, field name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ...

```
FETCH fields(#FP_EPTVAL) from_file(FPTABnma) with_key('VF_CURREN
```

```
Change #MyDescriptionField #FP_EPTVAL
```

I want to read through the currency codes and report on all transactions for each currency

*Use the logical view (nma) keyed by:

*Table name, field name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ...

```
SELECT *ALL from_file(FPTABnma) with_key('VF_CURRENCY' 'DESCRIP
```

(assuming that there is always a DESCRIPTN for a CURRENCY)

```
ENDSELECT
```

I want to read through the currency codes and report on all transactions for each currency, and I need to know both the exchange rate and the description

*Use the logical view (nma) keyed by:

*Table name, field name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ...

```
SELECT *ALL from_file(FPTABnma) with_key('VF_CURRENCY' 'DESCRIP
```

```
Change #MyDescriptionField #FP_EPTVAL
```

* Get a numeric cell value

* Use a second logical view to avoid confusing the pointer

```
FETCH fields(#FP_EPTNV) from_file(FPTABn2a) with_key('VF_CURREN
```

```
Change #MyExchangeRateField #FP_EPTNV
```

```
ENDSELECT
```

I want to report on all combinations of Currency and Department

*Use the logical view (nma) keyed by:

*Table name, field name, AKey1, Nkey1, AKey2, Nkey 2, Akey3, Nkey3 ...

```
SELECT *ALL from_file(FPTABnma) with_key('VF_CURRENCY' 'DESCRIP
```

```
Change #MyCurrencyCodeField #FP_EKEY1
```

```
Change #MyCurrencyDescriptionField #FP_EPTVAL
```

* Now read through all the departments

* Use another logical view (n2a) to avoid confusing the pointer

```
SELECT *ALL from_file(FPTABn2a) with_key('VF_DEPTAB' 'DESCRIP
```

```
Change #MyDepartmentCodeField #FP_EKEY1
```

```
Change #MyDepartmentDescriptionField #FP_EPTVAL
```

```
ENDSELECT
```

```
ENDSELECT
```

Note: It helps coding if for every table type there is a non-key field (e.g. Description) that must exist for every table entry.

I want to decode from a table with a numeric key

Say there is a table VF_POSTCODE keyed by #POSTCODE (numeric)

*Use the logical view (nmn) keyed by:

*Table name, field name, NKey1, Akey1, NKey2, Akey 2, Nkey3, Akey3 ...

```
FETCH fields(#FP_EPTVAL) from_file(FPTABnmn) with_key('VF_POSTCO
```

```
Change #MyDescriptionField #FP_EPTVAL
```

I want to decode from a table with a mixed key

Say there is a table VF_POSTCODE keyed by #POSTCODE (numeric) and #COUNTRY (alpha)

*Use the logical view (nmn) keyed by:

*Table name, field name, NKey1, Akey1, NKey2, Akey 2, Nkey3, Akey3 ...

```
FETCH fields(#FP_EPTVAL) from_file(FPTABLnmn) with_key('VF_POSTCO
```

Change #MyDescriptionField #FP_EPTVAL

I want to read departments in description order

*Use the logical view (val) keyed by:

*Table name, field name, Numeric Property Value, Alpha Property Value.

```
SELECT *ALL from_file(FPTABval) with_key('VF_DEPTAB' 'DESCRIPTN'
```

(assuming that there is always a DESCRIPTN for a department)

```
ENDSELECT
```

Using Assistants to handle more complex "codes"

The preceding sections deal with handling simple codes, decodes and prompting (things such as Currencies, Companies, Countries, etc). This approach is okay when working with small sets of relatively static information.

However in most commercial applications there are "codes" such as Customer Numbers, Order Numbers, Policy Numbers, Product Numbers.

These types of codes have several important characteristics:

- The information associated with them is often quite dynamic.
- They are used all over the application and cross business objects. For example, when creating a new Order references to the "codes" for Customers and Products are heavily used.
- They are used all day every day by the end-users. Using these "codes" and moving around within them is at the core of the application's functionality.

An interesting way to handle them involves the generic concept of an "Assistant".

An "Assistant" hangs around in a small window (or windows) ready to pop up and offer specialized assistance to the end-user at the appropriate time (somewhat like the one used in MS-Office applications).

The traditional F4=Prompt option used in many 5250 style applications is an example of a very specialized type of Assistant (i.e.: it pops up at the appropriate time and assistants the end-user to identify the "code" that they want to use.

However, in the Windows GUI world the concept of an assistant can be significantly extended to offer substantially more functionality to the end-user.

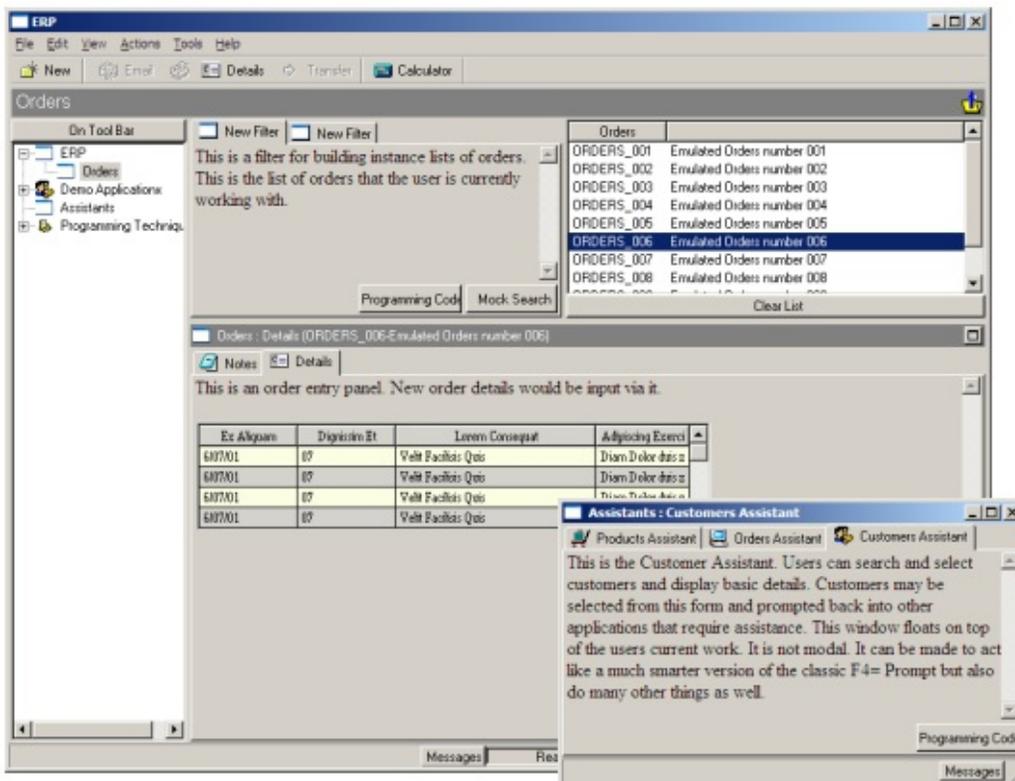
For example in a large order processing application there might be "assistants" for Customers, Products and Orders ... which are the three predominant business objects within the application.

In the Framework the assistant(s) could be made to appear to end-users in several ways:

Approach 1: Single Window – Multiple Assistants

Here the Framework designer has defined a single application called "Assistants" which has no filters and three full screen commands called "Product Assistant", "Order Assistant" and "Customer Assistant". "Product Assistant" is the default command.

When the end-user clicks on the "Assistants" application they immediately see a single window with 3 tabs (one for each assistant) like this (the actual content of the assistant windows has been omitted):

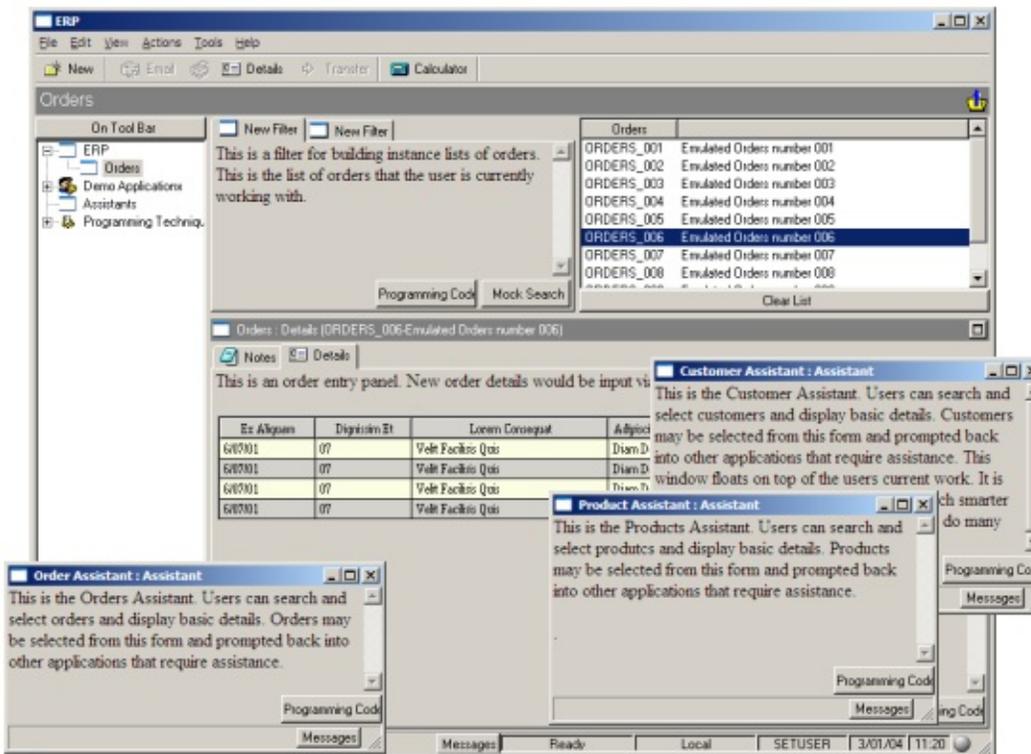


Note that the assistant window floats on top what the end-user is currently doing in the main Framework window. The assistant can interact with the application in the main window in any way that is desired. For example, selecting a customer in a Customer Assistant might cause the customer number to appear in the "Customer Number" entry field on the underlying Order Entry form (this is the classic F4=Prompt interaction done in a non-modal way).

Approach 2: Multiple Windows – Multiple Assistants

Here the Framework designer has defined a single application called "Assistants" which contains three business objects called "Product Assistant", "Order Assistant" and "Customer Assistant". Each has a single command (which is the default).

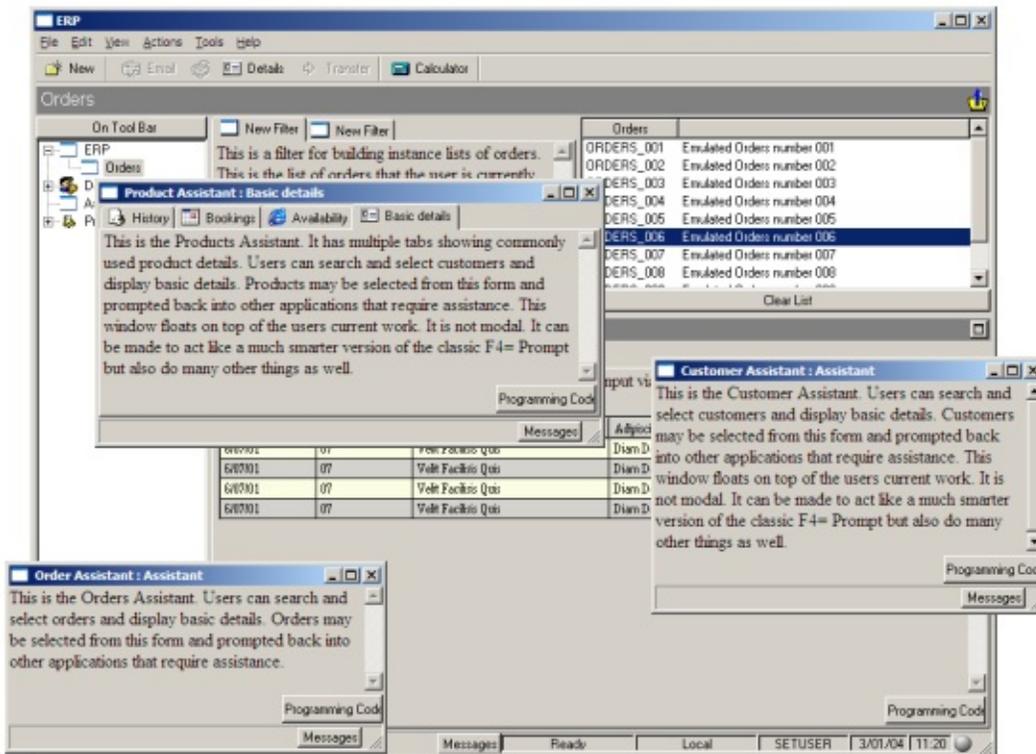
When the end-user clicks on "Assistants" they can select from "Product Assistant", "Order Assistant" or "Customer Assistant". By clicking on one of more the Assistants they can display each assistant in it's own window.



Note that the assistant window(s) float on top what the end-user is currently doing in the main Framework window. Here the end-user has all three assistants active together to help them input order details into the main form underneath.

Approach 3: Multiple Windows – Multiple Assistants (some Complex)

This approach is very like approach 2, except that selected assistants can be made more complex and have multiple tabs of information associated with them:



Here the "Product Assistant" has been made more complex and it has the tabs "History", "Bookings", "Availability" and "Basic Details" associated with it.

In using approach 3 it is important to remember that something like a "Product Assistant" primary role is to help users locate and display commonly requested product information only. It can be set up to be invoked from anywhere that product information is displayed anywhere else in your application.

You can get carried away with an Assistant and attempt to make it too powerful and too encompassing, to the detriment of the real "Product" business object in your application.

How are Assistants started?

Since assistants are just normal Framework or business object level command handlers they can be started from many different places directly by end-users:

- Option(s) in drop down menus
- Icon(s) on the toolbar
- Right Mouse menu option(s)

How and when they appear in these situations is definable by the Framework designer.

Programmatically you can also start an assistant by using the normal Switch operation.

How do Assistants interact with other parts of the application?

Assistants are free floating modeless windows.

They communicate with the rest of the application by signaling events to it.

The respond to the rest of the application by listening for events signaled by it.

This very simple architecture is flexible and extensible.

Here's some examples of simple even driven interactions:

Example 1 – A classic F4= Prompt style request

An order entry form is asking the user to input a customer number.

The user indicates that they would like to prompt the customer by pressing a button or an image (eg: a question mark).

The Order Entry form signals a "SelectCustomer" event and may optionally include details of information already input by the user (eg: partial customer name, number etc) as part of the "payload" associated with the event.

The Customer Assistant detects the "SelectCustomer" event and allows the user to search for the required customer. The "payload" associated with the event may be used to speed up or narrow the initial search. When the user locates and selects the required customer the Customer Assistant signals a "CustomerSelected" event and would include the customer number (and maybe their name) as the "payload" associated with the event.

The Order Entry form detects the "CustomerSelected" event and updates the details of the order being input with the selected customer details.

Example 2 – A more advanced non-modal prompt request

An Order Entry form is currently active.

The Customer Assistant window is also floating around.

The user goes directly to the Customer Assistant window and searches for a customer. When they select it the Customer Assistant signals a "CustomerSelected" event and would include the customer number (and maybe their name) as the "payload" associated with the event.

The Order Entry form detects the "CustomerSelected" event and updates the details of the order currently being input with the selected customer details.

In web applications particularly this is a more efficient way to perform prompting because it cuts out the number of interactions with the remote server that are required to complete the interaction.

Example 3 – Multiple Item Prompting

Event driven processing allows multiple item selections.

An Order Entry form is currently active.

The Product Assistant is window also floating around.

The user goes directly to the Product Assistant Window and selects four different products. When they have completed this selection the Product Assistant signals a "ProductsSelected" event. It would include a list of product numbers (and maybe product descriptions) in the payload of the "ProductsSelected" event.

The Order Entry form detects the "ProductsSelected" event and updates the details of the order currently being input with the all the products selected.

Example 4 – Some other uses for Assistants

Here's some ideas for extending the concept of an Assistant ...

- An all powerful Assistant. The preceding examples used 3 discrete assistants. There is nothing to stop you designing a single assistant that can handle multiple things. For example, a single assistant could easily be designed to signal "CustomerSelected", "ProductSelected" or "OrderSelected" (with appropriate payloads) depending upon its interaction with the user.
- A "Currency Calculation Assistant" that performs currency calculations and broadcasts "CurrencyCalculationPerformed" events.
- An "Alerts and Messages Assistant" that polls a server application periodically for alerts and messages that are relevant to the current user. When one is detected it switches to a Framework level command to display the details to the user.
- A "Chat and Messages Assistant" that allows the end-user to create and send messages to other application users in a controlled manner. This would probably be linked to the preceding example in some way (or simply an extension of it).
- A "Server" Assistant that allows the end-user to display and manipulate server side things that he/she has initiated (eg: Reports, Output queues, Message Queues, etc).

Using Unicode Data with the Framework

The Framework can handle Unicode data for certain parameters in certain methods.

For example, all the methods for reading and writing data to/from the instance list can handle Unicode data for the Visual Identifiers and AColumns.

The general rules are:

Writing Data

When writing data to the Framework, you can continue to use the same parameter, and use either an Alphanumeric field or a Unicode (NChar NVarChar) field as input.

For example both these statements are valid:

```
#avListManager.AddtoList AColumn1(#MyAlphaField)
#avListManager.AddtoList AColumn1(#MyUnicodeField)
```

Reading Data

When reading Unicode data from Unicode-enabled parameters, you must use the new u version of the parameter if you want the Unicode data to be displayed correctly.

For example if reading Unicode-data that is outside the native character set, this statement is not valid:

```
#avListmanager.GetSelectedInstance AColumn1(#MyUnicodeField)
```

But this one is OK:

```
#avListmanager.GetSelectedInstance AColumn1u(#MyUnicodeField)
```

However, if you are not reading Unicode data that is outside the native character set, all these statements are OK:

```
#avListmanager.GetSelectedInstance AColumn1(#MyUnicodeField)
#avListmanager.GetSelectedInstance AColumn1(#MyAlphaField)
#avListmanager.GetSelectedInstance AColumn1u(#MyUnicodeField)
```

So pre-existing logic will continue to run ok.

Relationship Handlers

If you are using a relationship handler to build tree-style instance lists, and you want to add Unicode data to the instance list, you must use a reusable part as the relationship handler, not a function.

Examples

See shipped examples DF_T35* and DM_T35* for examples of using Unicode with filters, command handlers, relationship handlers and snap in instance lists.

Note about VLF Tools and Extensions

If you have created VLF tools or extensions that read the internal VLF object model, you may need to modify and/or recompile them. Some of the VLF's internal properties have been changed to be Unicode and you may need to add `.asNativeString` to your references.

Also see:

[Unicode Data in Code Tables](#)

[Unicode Data in Virtual Clipboard](#)

[Unicode Data in Tracing](#)

Unicode Data in Code Tables

Unicode data can also be used in code tables if a reusable part data handler is used. Set the data type of the table column to 'U' in the Code Table definition tab of the Design Code Tables form.

A default reusable part table data handler, UF_TDH01 is shipped with the Framework. If UF_TDH01 is used as a table's data handler, data will be written and read from the shipped code table files FPTAB (main data file) and FPTABU (Unicode value file). Custom Reusable part data handlers can also be created and used with code tables.

If a custom data handler reusable part is created, it must extend VF_AC024, the Table Data Handler ancestor class, and redefine the methods included in the ancestor class. The default behavior can be optionally included by calling the ancestor methods in the redefined methods first. See comments in UF_TDH01 for more details.

Unicode Data in Virtual Clipboard

To save Unicode values use the `avSaveValue` method as you normally would but pass Unicode values to the `FromAValueU` parameter:

```
#AvFrameworkManager.avSaveValue WithID1(PRIVATE) WithID2(*COMPC
```

Similarly when retrieving Unicode values:

```
#AvFrameworkManager.avRestoreValue WithID1(PRIVATE) WithID2(*COM
```

Unicode Data in Tracing

The Framework manager's basic tracing service is now Unicode enabled. Unicode strings can be traced and will present in the Tracer form and in the saved trace files. Unicode strings and Unicode enabled fields can be passed to the tracing service using the Event parameter or the AValue parameters.

For example:

```
#AVFRAMEWORKMANAGER.avRecordTraceAValue Avalue(#VF_elurlu)  
Component(#COM_OWNER) Event('Trace the value of a Unicode field,  
#VF_elurlu - ')
```

End-user Help (F1)

This section describes how you create and display:

[Help Text for Windows Applications](#)

[Help Text for Web Applications](#)

[Disabling or diverting the F1=Help key in Framework applications](#)

Help Text for Windows Applications

Help for Windows Framework applications is handled through the normal LANSAs field and component based help text facility. See the sections on Help Text in Visual LANSAs Components – A Developer's View and the iSeries User Guide for details about creating and displaying help text for Windows applications.

Help Text for Web Applications

The Framework optionally provides F1=Help support to your web browser applications.

It does this by trapping F1=Help requests made by the user and then invoking a WEBEVENT function named UFU0002 in process UF_SYSWB.

Alternatively the shipped UF_SY002 offers a basic example of how to use a WAM to display application help. Refer to the Web Details tab in Framework Properties:

Web Help Text	
<input type="radio"/> Use a WEBEVENT for Help	<input checked="" type="radio"/> Use a WAM for Help
Web Help Process/WAM Name	UF_SY002
Web Help Function/Webroutine Name	onhelp
Web Help Window Features	width=400,height=200,directories=no,toolbar=no,me

Developers may modify the default behaviour by creating their own version of UF_SY002 (with a different name) and cause it to display a CHM document or even a specific section of a CHM document.

To activate F1=Help processing in your WAM filters or command handlers:

- Use WAMHELP=Y on the URL you use to start your Framework application. This activates WAM Help.

Disabling or diverting the F1=Help key in Framework applications

The F1 key is special in all Visual LANSA applications. It invokes the online help, and it is almost impossible in any Windows application created with any tool to use it for any other purpose.

However, you can disable it, or capture it and use it in a different way (for example to display a CHM document).

The following example demonstrates how to capture and disable the F1=Help key. It is a customized copy of the shipped user entry point form UF_EXEC.

To try it, create a form with a name different to UF_EXEC and paste in the following code.

See the notes at the end of this code for more information about creating customized entry point forms.

```
*
=====
*
* Component   : XX_EXEC
* Type       : Form
* Ancestor   : VF_AC006
*
*
=====
*
* PLEASE NOTE: This component is a COPY of the shipped version.
*               You may choose to modify it. Refer
*               to the end of this component for more details about making your
*               own version of this component.
*
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #VF_AC006 *implements
#Prim_App.IHelpHandler) CLIENTWIDTH(771) LEFT(153) TOP(32)
WIDTH(779)
DEFINE_COM CLASS(*ANCESTOR) NAME(#BROWSER) WIDTH(247)
DEFINE_COM CLASS(*ANCESTOR) NAME(#COMMANDHANDLER)
WIDTH(578)
```

```
DEFINE_COM CLASS(*ANCESTOR) NAME(#COMMAND_PANEL)
WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#INTRO) WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#INTRO_PANEL)
WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#RIGHT_PANEL)
WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#SELECT_PANEL)
WIDTH(247)
DEFINE_COM CLASS(*ANCESTOR) NAME(#TOP_PANEL)
WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#WORK_PANEL)
WIDTH(578)
DEFINE_COM CLASS(*ANCESTOR) NAME(#STATUS) LEFT(239)
DEFINE_COM CLASS(*ANCESTOR) NAME(#TOOLBAR) WIDTH(767)
DEFINE_COM CLASS(*ANCESTOR) NAME(#IDENT_PANEL)
WIDTH(767)
DEFINE_COM CLASS(*ANCESTOR) NAME(#MAIN_PANEL)
WIDTH(767)
DEFINE_COM CLASS(*ANCESTOR) NAME(#IDENT_BUTTON)
LEFT(746)
DEFINE_COM CLASS(*ANCESTOR) NAME(#CURRENT_OBJECT)
WIDTH(467)
DEFINE_COM CLASS(*ANCESTOR) NAME(#APPLICATION)
WIDTH(185)
DEFINE_COM CLASS(*ANCESTOR) NAME(#LEFT_PANEL)
WIDTH(185)
DEFINE_COM CLASS(*ANCESTOR) NAME(#STATUS_BAR)
WIDTH(771)
DEFINE_COM CLASS(*ANCESTOR) NAME(#MiniFilter) WIDTH(467)
DEFINE_COM CLASS(*ANCESTOR) NAME(#Right_panel_Main)
LEFT(189) WIDTH(578)
```

```
*
```

```
=====
* Method Routines
```

```
*
```

```
=====
```

MthRoutine uInitializeFramework Options(*Redefine)

* Set up for end user mode

Set #Com_Owner iDesignMode(FALSE)

Set #Com_Owner uAdminMode(FALSE)

* Set to start up image name

Set #Com_Owner uStartupImage(#uf_im001)

* Nominate the XML file containing the Framework design

Set #Com_Owner uSystemXMLFile('vf_sy001_system.xml')

uSystemXMLChoice('vf_sy001_system_choice')

EndRoutine

Mthroutine Name(ProcessHelpRequest) Options(*Redefine)

* Define_Map For(*input) Class(#prim_objt) Name(#Requestor)

Pass(*by_reference)

* Define_Map For(*input) Class(#prim_alph) Name(#Tag)

* Define_Map For(*input) Class(#prim_bo1n) Name(#Handled)

* Add logic here to determine if F1 is ignored or not.

* If Handled is true this will stop the help request going to the normal VL help facility.

Set Com(#Handled) Value(True)

Endroutine

*

=====

* MAKING YOUR OWN VERSION OF THIS COMPONENT

*

=====

*

* It is not recommended that you create development or design entry points for your

* framework. Simply use the shipped ones and use the 'Save As' option to create

* different frameworks. Then only create administrator and user entry points as

* required for production users, locking them into a specific XML file and never

* allowing the end user a framework XML file choice.

*

* To create your own User entry point do the following:

*

* -> Create a VL form with your chosen entry point name (eg: MYEXEC).

*

* -> Copy the code from UF_EXEC into your new form. Initially this will cause

* errors to be displayed.

*

* -> Change the ancestor of your new form to VF_AC006.

*

* -> The copied code should contain a method routine that will look like this:

*

```
* MthRoutine uInitializeFramework Options(*Redefine)
```

```
* Set #Com_Owner iDesignMode(FALSE)
```

```
* Set #Com_Owner uAdminMode(FALSE)
```

```
* Set #Com_Owner uStartupImage(#uf_im001)
```

```
* Set #Com_Owner uSystemXMLFile('vf_sy001_system.xml')
```

```
* EndRoutine
```

*

* This code defines whether this entry point should allow application

* design (iDesignMode), whether the administration of users and servers

* should be allowed (uAdminMode). It also defines what the startup bitmap

* to be shown is (uStartupImage) and the name of the XML file containing

* the framework design.

*

* -> Change these properties as desired and then compile and test your
* entry point. You should not make any other changes to the logic
* in your entry point.
*
* -> Optionally include a uSystemXMLChoice file name to allow the user to
* select which framework should be opened from a list contained in the
* specified file.
*
* -> In design mode entry points only, optionally add the
uSystemXMLSaveAs
* property to indicate the designer can save the framework XML file with a
* different name.
*
* -> If you enroll a bitmap into the LANSAs repository, say, your company
* logo under the name #MYLOGO then changing the line:
*
* Set #Com_Owner uStartupImage(#uf_im001)
*
* to:
*
* Set #Com_Owner uStartupImage(#MYLOGO)
*
* will cause your logo to be presented while the framework is starting.
*
* -> UF_DESGN, UF_DEVEL, UF_ADMIN and UF_EXEC are designed to
act as framework
* entry points only. You should not try to use UF_DESGN, UF_DEVEL,
UF_ADMIN and UF_EXEC
* (or any copied version of them) inside any framework in any way.
*

End_Com

Designing Filter and Command Handler Layouts

When you are implementing filters or command handlers you will need to understand how to design different form layouts. The recommended approach to designing forms is:

- Use the Code Assistant to generate the initial code for your filter or command handler. Alter this code as required to add any additional functionality to your filter or command handler.
- Next, compile your filter or command handler. This will cause a basic form layout to be created for you.
- Continue to use the basic form layout until you are satisfied that your filter or command handler is functionally complete.
- When your filter or command handler is functionally complete, change the form layout to be exactly what you want. This is called fine detailing. Do this only when your filter or command handler is functionally complete.

How you fine detail your form layouts depends on the type application you are working with:

WINDOWS: Use the Visual LANSA Form Editor just as you would for any other Windows application.

WAM: Use the WAM Form Editor just as you would for any other WAM based application.

Programming Tips

Component Names and Identifiers

Advanced Enter Key Handling in VL applications

Possible Technique for Handling "New" in VLF And Ramp Application Designs

Component Names and Identifiers

LANSAs components have a name, which may be longer than 8 characters, and an identifier.

When working with the Framework, use the identifier when specifying the components or functions to be used as filters, command handlers, imbedded interface points, etc.

However, you can prompt and search for plug-in components using their long name. When you select a component, its identifier is returned and saved in the XML schema.

Advanced Enter Key Handling in VL applications

The Enter key is often used to indicate the initiation of some activity (eg: to perform a search, to save an update, etc) in VL applications.

Typically the ButtonDefault() property is used on push buttons such as "Search" or "Save" so that the Enter key causes a virtual click event to be issued against the button.

In complex VL forms containing many different reusable parts the use of the ButtonDefault() property may become problematic for two reasons:

- Only a single button can be the default at any point in time on a Windows form.
- There is no ability to use program logic to decide what the Enter key actually means and how it should be handled in different contexts.

The most powerful solution to handling the Enter key differently in different places on complex forms is be solved by using the KeyPress event. This allows the VL program to specifically trap the Enter key and then respond to it in different ways.

Consider the following modifications made to the logic in the filter DF_FILT1 example shipped with the VLF.

1. No button has ButtonDefault() specified

This prevents any confusion about which button the Enter key should be directed to.

2. The user may initiate an Employee name search in two ways

```
Evtroutine Handling(#Surname.KeyPress) Keycode(#KeyCode)
if ('#KeyCode.Value = Enter')
If *SEARCHOK
    Execute Search
Else
    Use Message_box_show (ok ok Error *Component 'Enter a search name')
```

```

Endif

Endif

Endroutine

*
* Search button pressed
*

Evtroutine Handling(#Search_phbn.Click)

Execute Search

Endroutine

```

The first event routine handles the use of the Enter key within the Surname field. If the user presses Enter then either a search is performed or an error message is displayed (Condition *SEARCHOK is true if field #Surname is non-blank).

The second routine handles a specific click of the Search button.

Both routines share a common SEARCH subroutine to perform the search logic.

Note that where multiple fields (or other visual controls) are used you can handle them all through a single routine like this:

```

Evtroutine Handling(#Surname.KeyPress #GiveName.KeyPress
#ComboBox1.KeyPress etc etc)
    Keycode(#KeyCode)

```

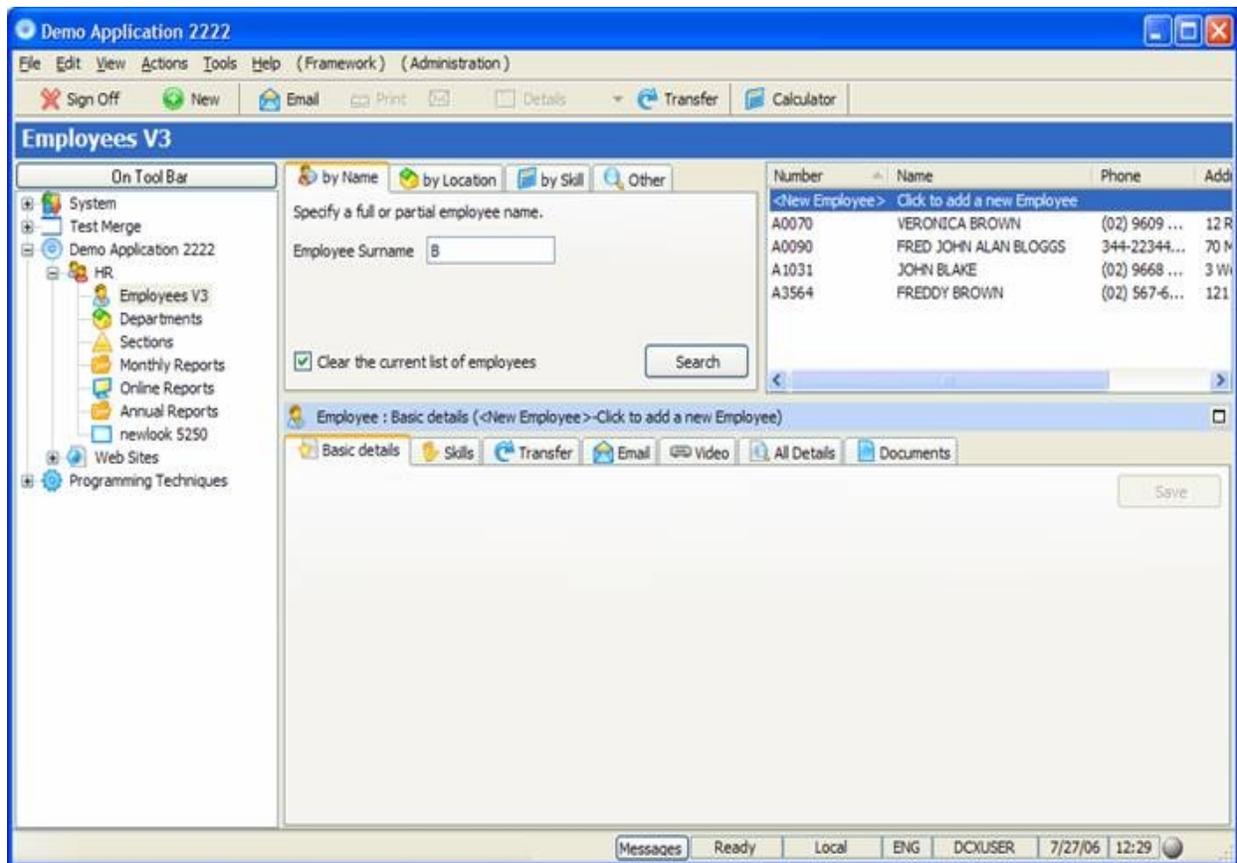
Possible Technique for Handling "New" in VLF And Ramp Application Designs

This VLF design approach:

- Maximizes command handler Visual LANSA code reuse
- Allows you to easily pipeline users using wizard style processing for new objects (eg: New Orders, New Customers, etc).

Any normal Framework filter can add a special type of "New" object to the instance list (even at entry, before any searching is done):

For example:



Here a demo system shows an Employee with number <New Employee> and Name Click to Add a New Employee.

When the instance list entry is clicked the Basic Details command handler tab for <New Employee> is invoked. It sees something special in the instance list

key (for example "<NEW> in AKey1) and knows that it is being invoked to input a new employee's basic details, rather than display the details of an existing employee and it changes its behavior accordingly.

Basic Details command handler might also decide when handling a <NEW> employee that when the Save button is clicked that the user is automatically driven to the Skills tab, which might then decide to drive them to the Documents tab.

In other words the three tabs Basic Details, Skills and Documents act like a New Employee wizard that drives (or pipelines) the user creating a new employee through the required tabs. The tabs could even change their button captions to Next and Previous when handling a <NEW> objects, instead of using the more traditional OK and Save they use for updating existing employee details.

Quick Find Override Feature

The Quick Find box is a dialog that appears on the top right of the VLF, if enabled. It is enabled in the Framework Details tab by setting the Search Field Width to a non-zero value:



As the user types in a search string, a list of all the business objects that have a caption that contains this search string are displayed, and the user can click on any of them, to immediately switch to that business object.

Why Override the Way Quick Find Works?

The standard behaviour is for all the business objects that contain the search string to appear in the list for selection.

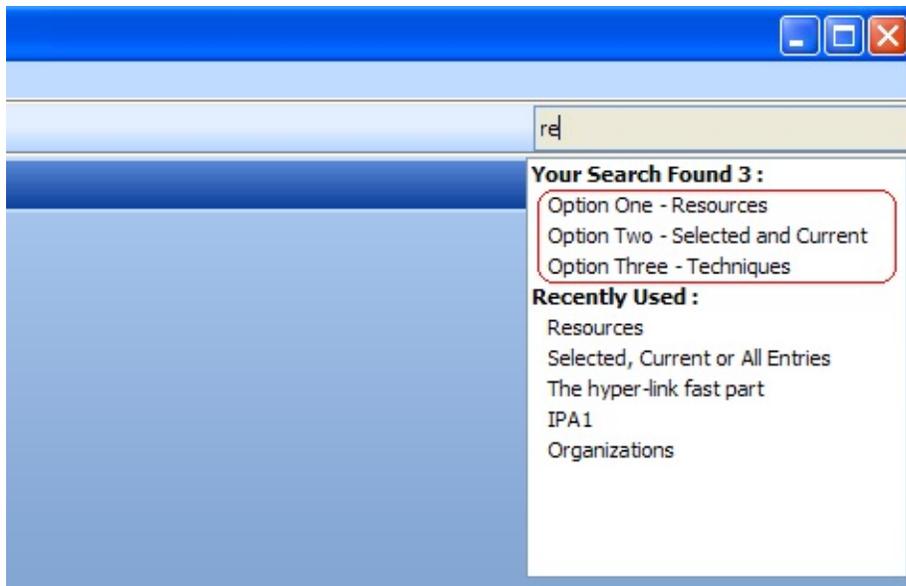
For very large frameworks with many business objects this may not be what the user wants.

Instead, they may want a shorter list of their commonly used options. And they may want a specific switch to a command within a business object when the user clicks that option.

They may want a shortcut value to take them to a particular business object, even though the shortcut is not present in the business object's caption.

What Does Overriding Quick Find Do?

You can override what appears when the user searches using the quick Find Box, and what happens when they click on one of the entries displayed.



The marked area in the picture above is what can be controlled. (The Recently used options behave as they always have. They can be deactivated if not required.)

Setup

To override the search options, modify your version of the IIP component UF_SYSTEM.

Step 1

Start by activating the override feature by setting PavQFUserOverride to True in this routine:

- * To turn on Quick Find list overriding change FALSE to TRUE
- * in this property get routine ...

```
Ptyroutine Name(GavQFUserOverride) Options(*Redefine)
Set Com(#PavQFUserOverride) Value(True)
Endroutine
```

(This also stops the framework from searching its own list of business objects and applications.)

Step 2

Control the complete list of all possible options that the user can search through by adding your options in this routine:

- * Use this method to load the Quick Find list with your own values
- * You must set PavQFUserOverride to true for this to be active

```
Mthroutine Name(avQFLoadSearchList) Options(*REDEFINE)

Invoke #avQuickFind.AddSearchListEntry Text('Option One - Resources')
GUID('GUID1')
Invoke #avQuickFind.AddSearchListEntry Text('Option Two - Selected and
Current') GUID('GUID2')
Invoke #avQuickFind.AddSearchListEntry Text('Option Three - Techniques')
GUID('GUID3')

Endroutine
```

For each possible entry, add the text to be displayed (82 characters), and a

unique identifier (32 characters).

When the user enters a search value, the option text will be searched to see if it contains the search string, and will appear as an option for the user if it does.

Step 3

Control what happens when the user clicks on an option using this routine:

What you do could be anything, but most commonly would be a switch.

```
* This routine receives the selection that the end-user made in the Quick Find list.
```

```
* This routine determines what happens when the user selects an entry from the Quick Find list
```

```
* You must set PavQFUserOverride to true for this to be active
```

```
Mthroutine Name(avQFActionSelection) Options(*REDEFINE)
```

```
define #vf_wkGUID reffld(#VF_ELIDN)
```

```
change #vf_wkGUID #GUID
```

```
Case OF_FIELD(#vf_wkGUID)
```

```
when '= GUID1'
```

```
* Resources
```

```
Invoke #avFrameworkManager.avSwitch To(BusinessObject)
```

```
Named(DEM_ORG_SEC_EMP) Execute(Details) Caller(#Com_Owner)
```

```
when '= GUID2'
```

```
* Selected and Current
```

```
Invoke #avFrameworkManager.avSwitch To(BusinessObject)
```

```
Named(BD8D1D64689F468E8FA84DA2DA1A87FA) Caller(#Com_Owner)
```

```
when '= GUID3'
```

```
* Programming Techniques application
```

```
Invoke #avFrameworkManager.avSwitch To(Application)
```

```
Named(78A26552E6E94627BBCC4DF3A248CD1F) Caller(#Com_Owner)
```

```
endcase
```

```
Endroutine
```

Step 4 (Optional)

You can cause the complete list of options to be rebuilt.

Normally the complete list of all possible options that the user can search through is built only once. However, it is possible to indicate to the QuickFind component that it should rebuild the list of possible options the next time the user modifies the search string.

This can be done from anywhere that can access #uSystem (e.g. a command handler or filter), with an instruction like:

```
set #USYSTEM.uUserIIP avQFRebuildList(true)
```

If setting this property to true and using the override feature, it is necessary for the avQFLoadSearchList IIP routine to reset avQFRebuildList to false when it has finished loading.

Advanced Topics

Using your Visual LANSA Framework in Direct-X mode

Considerations for ISVs

Deploying a Framework Version

Users, Groups and Security

Server Profile Management and Issues

Multilingual Application Issues

Imbedded Interface Points (IIPs)

Custom Properties

Writing queries over Visual LANSA Framework objects

Using your Visual LANSA Framework in Direct-X mode

When you start a version 13 Visual LANSA application, you have a choice of these Render Types:

W Win32

X DirectX

M Mixed W32 and DirectX

However, the Visual LANSA Framework can only be run using Render Types W and M.

[Win32](#)

[Mixed Mode](#)

Win32

If you start the UF_DESGN/UF_DEVEL/UF_ADMIN/UF_EXEC form using Render Type W, the application will run in Win32 mode, which is closest to the way it has always run prior to Version 13 of Visual LANSa.

All parts of the VLF will run in this mode, even any of your own snap in components with the RenderStyle set to Direct-X.

Mixed Mode

If you start the UF_DESGN/UF_DEVEL/UF_ADMIN/UF_EXEC form using Render Type M, most of the Visual LANSAs Framework application will run in Win32 mode, but the instance list, and any snap in instance lists will run in Direct-X mode.

[Snap In Instance Lists](#)

[Snap In command handlers and filters](#)

Snap In Instance Lists

If you have any snap in instance lists, they will now be running in Direct-X mode, so you should test carefully that your snap in instance lists still behave the way you want. Direct-X mode instance lists will look different and may behave differently.

In particular, in Direct-X the list's current item changes as the user moves the mouse over it, and the source fields of the list columns also change. Previously (Win32 mode) the current item changed only when the user clicked on an item.

Also check that your instance list resizes correctly. This can be a problem if your snap in list has W32 controls (including activeX and the List View) that are not directly sized by the parent object's layout manager.

Snap In command handlers and filters

Your snap in command handlers and filters and any RAMP command handlers will run in Win32 mode as they did previously.

You have the option of changing the code of your snap in command handlers and filters, and setting the component's RenderStyle to DirectX, and recompiling. If you do so, and the VLF is running in Mixed mode, they will run as DirectX components. You should be extremely cautious and test very carefully when doing this.

An important difference is that in DirectX mode, when the user mouses over a list/grid, the source fields of the columns change automatically.

If your command handler uses a list and a details panel that changes when the user clicks on a list entry, you may find that some fields on the details panel are now changing as the user mouses over the list. If the user then saves the data on the details panel, they may be saving a mixture of altered and unaltered field values.

Another important difference is in the sizing of W32 objects (in particular ActiveX controls) when their size is not controlled directly by a layout manager.

Another important difference is that bringing a panel or control to the front does not hide other controls or panels behind it.

Another difference is in the font used by DirectX which may occupy more space than the W32 font.

This is not a complete list of all the differences between DirectX and the standard W32 mode.

Be cautious if you decide to change a filter or command handler to use DirectX mode. You may well need to redesign it. Do not under any circumstances change over all your filters and command handlers without understanding the differences between DirectX and Win32, and thoroughly testing as you change.

Considerations for ISVs

If you are an ISV developing Framework based applications for sale, there are some additional Framework issues you should consider. These primarily revolve around name space contention.

For example, a standard Framework will reference components UF_ADMIN, UF_EXEC, UF_DESGN, UF_DEVEL, UF_SYSTM and UF_IB001. If a customer has Frameworks supplied by you and from another source, and attempts to use them in the same LANSAs partition, then there will be contention for the namespaces of these UF_ components.

Please refer to [Deploying a Framework Version](#) for more details of these issues and how to overcome them.

Deploying a Framework Version

Once the development/testing cycle has finished and it is time to ship the executable version you will have a single Framework definition.

Let's assume the final Framework definition XML file is Ship.xml. This is the only Framework that the end-users will execute.

Because end-users must not have a choice of Frameworks, you must create your own versions of the available entry point forms. However, it is not recommended that you create development or design entry points for your Framework. Simply use the shipped ones and use the 'Save As' option to create different frameworks. Then only create administrator and user entry points as required for production users, locking them into a specific XML file and never allowing the end-user a framework XML file choice.

1. Create the required customized entry point forms to your Framework. These might be named, say, SHIPADM and SHIPUSR.
2. In the uInitializeFramework routine of all the forms change this line to disable the SaveAs functionality:

```
Set Com(#Com_Owner) Usystemxmlfile('vf_sy001_system.xml')  
Usystemxmlchoice('vf_sy001_system_choice')  
Usystemxmlsaveas(TRUE)
```

To:

```
Set Com(#Com_Owner) Usystemxmlfile('ship.xml')  
Usystemxmlsaveas(FALSE)
```

3. Compile SHIPADM and SHIPUSR.
4. You now have a completely independent Framework (named SHIP). It is designed by executing form UF_DESGN, administered by executing SHIPADM and you execute applications via SHIPUSR.

[Naming Space Considerations for Managing Different Frameworks](#)

Refer to [Visual LANSA Framework Deployment Check Lists](#) for detailed information on how to deploy your application.

Naming Space Considerations for Managing Different Frameworks

Naming space issues may arise when managing independent Frameworks, especially when you install your Framework at a site that is using Framework(s) supplied by other software vendor(s).

If you have different Frameworks in use in the same LANSAs partition, a name space conflict may arise with:

Framework Entry Points: Multiple independent Frameworks cannot have the same entry point names (for example UF_DESGN, UF_DEVEL, UF_ADMIN, UF_EXEC).

Filters and Command Handlers: You cannot have two different command handlers with the same name in the same LANSAs partition. When creating a new Framework you should assign it its own unique naming standard for filters and command handlers.

You cannot have two versions of these **shipped Framework programs** in the same partition:

- UF_IB001 icon and bitmap enroller.
- UF_SYSTM User Imbedded Interface Point (IIP).
- UFU0001 (process UF_SYSBR) sign-on program for Web browser applications.

Look at the source code of these programs for details of how to create a Framework unique version.

Users, Groups and Security

The Framework has a convenient, optional security system with user profiles, passwords, and authorities to objects in the Framework. This system offers a moderate level of security and it can interface with your own security system if need be.

It is not and cannot be a one stop shop for security due to the nature and variety of environments it operates with.

Framework security is optional. If you want to, you can switch off Framework security and just use the server's native security (the http server security for web users). In this way you can still control who accesses your application, but you won't be able to control what they can do within the Framework.

[Basic Framework Security](#)

[Five Things You Need to Know About Basic Framework Security](#)

[Advanced Options](#)

[Frequently Asked Questions \(Users and Security\)](#)

[Some Scenarios](#)

[More Information](#)

[Creating Web Interface for Maintaining Users and Authorities](#)

Basic Framework Security

The Framework can maintain its own user profiles, their passwords, and their authority to objects in the Framework.

It stores this information in the database files VFPPF06 and VFPPF07.

This is in addition to the server's own native security system of user profiles and passwords

[Set Up Server or Servers](#)

[Activating Basic Framework Security](#)

[Maintaining Users](#)

[Maintaining User Authorities](#)

[Maintaining User Groups](#)

Set Up Server or Servers

If Framework applications are going to be run as Web applications that use Framework-defined users and authorities or if user and authority details are to be maintained on a remote server, a server profile must be correctly set up in the Framework.

Server profiles are set up using the (Administration) menu, (Servers...) option.

- Select the New button and complete the Identification tab
- Complete the Server Details tab. Note that the server name must correspond to an entry in the LANSAs Communications Administrator.
- Close and save the Framework.

The screenshot shows the 'Servers' dialog box with the following configuration:

Sequence	Server
1	VLFPGMLIB
2	Local Test
3	My IBMi

Server Details Tab:

- Server Type: LANSA for System i
- Server Name: My IBMi
- Partition: TST
- Client-Server Translation Table: *JOB
- Server-Client Translation Table: *JOB
- Selection Block Size: 50
- Selection Limit: 10000
- Execution Priority: 20
- Server Overrides: (empty)
- Server IIP function to validate sign on: UFU0005

Identification Tab:

- DBCS Capable:
- Commitment Control:
- Divert Locks:
- Partition is enabled for RDMLX:
- Use Windows Credentials:
- Upper and Lower Case Password:
- Show on Connect Dialog:

Recover Connection Section:

- Attempt automatic session recovery:
- Time interval between checks of connection status (seconds): 30
- Check connection before executing commands:
- Check connection before selecting applications and business objects:
- Action to take when session cannot be recovered: Notify and allow retry
- Check connection using function: UFU0004

Buttons: New, Delete, LANSAs Comms. Admin, Close

Activating Basic Framework Security

Activation of this facility is done using the (Framework) menu, (Properties...) option, on the User Administration settings tab.

- Select the Use Framework Users and Authority setting. (Do not close and save the Framework yet)
- Select the Store Users in DBMS Tables VFPPF06/07 setting.
- Set End-users must sign on to this Framework to In both MS-Windows and Web Browser Applications
- Choose between Users Sign on Locally to Use the Framework or Users Sign on to a Remote Server to Use the Framework. Choose Users Sign on to a Remote Server to Use the Framework if Framework applications are going to be run as Web applications off a remote server or user and authority details are to be maintained on a remote server.
- Now close and restart the Framework

Framework

- Identification
- Custom Properties
- Visual Styles
- Icons
- Startup
- Commands Enabled
- Framework Details
- Command Display
- Export Design
- Help About
- Web/RAMP Details
- Developer Preferences - Web Server 1
- Developer Preferences - Web Server 2
- User Administration Settings
- Instance List Relationships Summary

Authority Settings

Use Framework Users and Authority

Store Users in DBMS Tables VFPFF06/7

Name of User Set to be Used:

Import Users Imbedded Interface Point:

Sign on Settings

End Users must Sign on to this Framework:

Users Sign on Locally to Use the Framework

User Can Change Own Password

Maximum Signon Attempts Allowed:

Users Sign on to a Remote Server to Use the Framework

Users May Work Offline if the Remote Server Is Not Available

-If Maximum Allowed Sign on Attempts Exceeded-

Advise User with a Message

Framework Fatal Error

Sign on Form Settings

Launch Button Caption: (ENG)

Launch URL (Windows):

Launch URL (Web / .Net):

Windows

Image / Web Page to Display:

Alignment of Image on Form:

Height:

Width:

Timeout Settings

Inactivity Log on timeout (minutes):

Inactivity Log off timeout (minutes):

Close

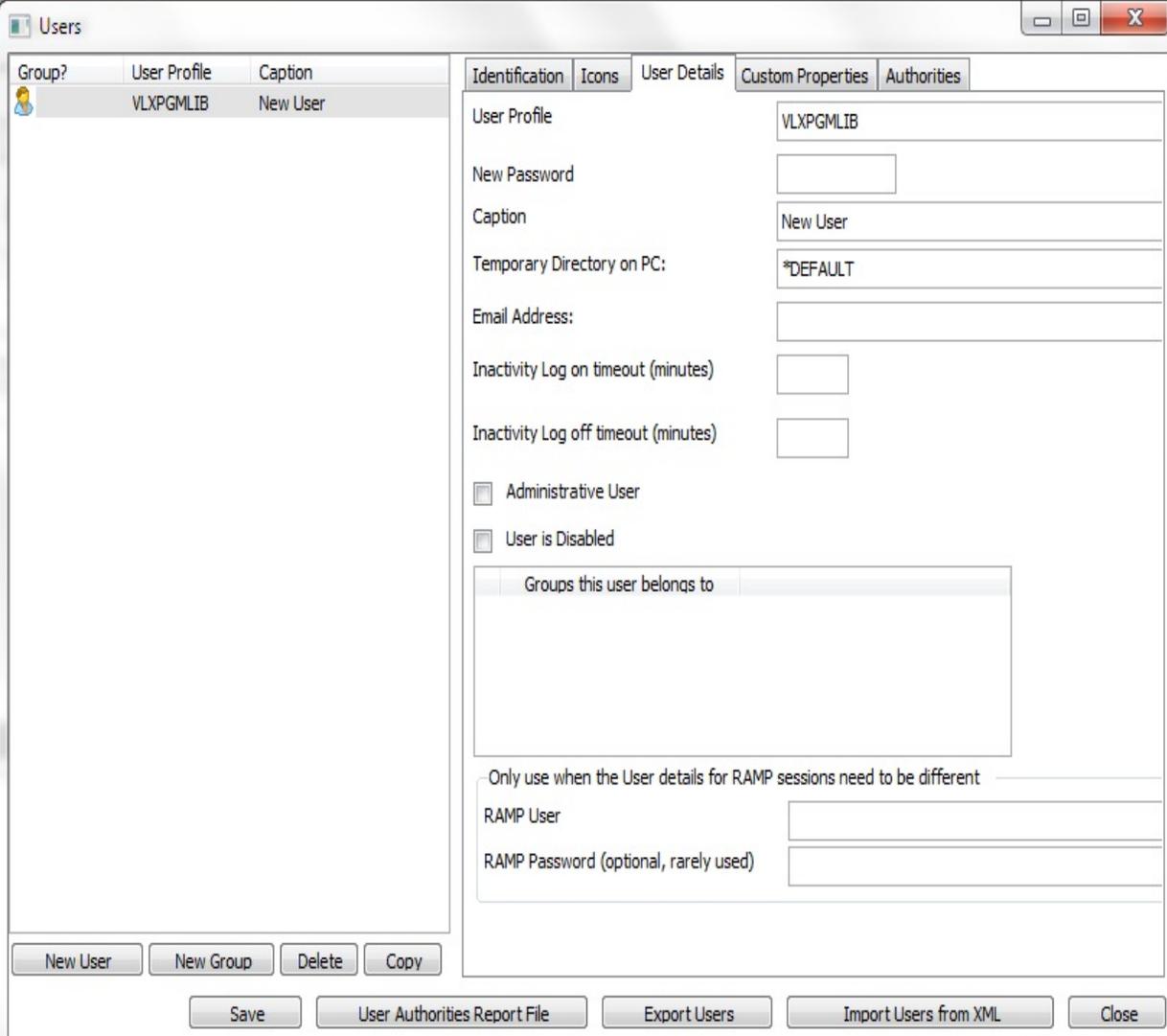
Maintaining Users

Now that Framework security is activated you can maintain users.

Start the Framework in design mode.

Go to the (Administration) menu --> (Users...), User Details tab.

You can use the "New User" button to create a user.



The screenshot shows a window titled "Users" with a tabbed interface. The "User Details" tab is active, showing the configuration for a user named "New User". The user profile is "VLXPGMLIB". The "New Password" field is empty. The "Caption" is "New User". The "Temporary Directory on PC" is set to "*DEFAULT". The "Email Address" field is empty. The "Inactivity Log on timeout (minutes)" and "Inactivity Log off timeout (minutes)" fields are empty. There are two unchecked checkboxes: "Administrative User" and "User is Disabled". Below these is a section for "Groups this user belongs to" which is currently empty. At the bottom, there are fields for "RAMP User" and "RAMP Password (optional, rarely used)", both of which are empty. The window has a standard Windows-style title bar with minimize, maximize, and close buttons. At the bottom of the window, there are several buttons: "New User", "New Group", "Delete", "Copy", "Save", "User Authorities Report File", "Export Users", "Import Users from XML", and "Close".

The most important setting for a user is the user profile. This value is used to identify a user. Both the Framework and the server identify the user using the user profile.

The password is used to validate a user's web sign on or local sign on.

You can press F2 on each setting to get contextual help

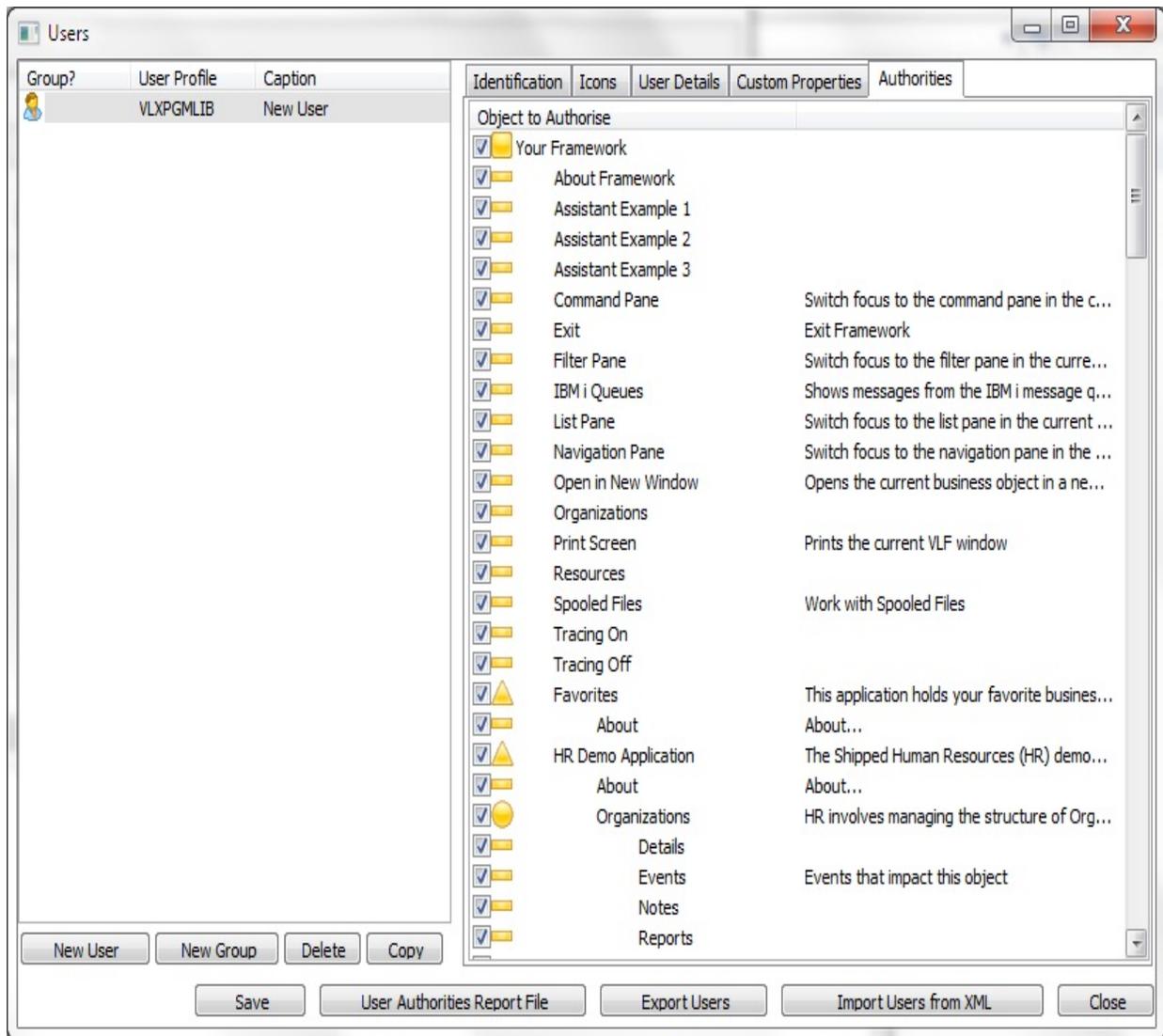
The first user you want to create is probably the administrator. Once this user is

defined to the Framework, this user can maintain other users by starting the Framework in administration mode.

You can copy the properties and authorities of the selected User or Group into a new one using the Copy button.

Maintaining User Authorities

You can maintain a user's authority to Framework objects on the Authorities tab:



If you uncheck a Framework object for a user, the user will not be able to see that object (application / business object / command) when they use the Framework in execute or administrator mode.

Maintaining User Groups

The Framework allows you to create user groups. You can assign Framework authorities group by group instead of individually.

A user can be a member of several groups, or one, or none. The user gets authority to anything they are individually authorised to as well as anything that any group they belong to is authorised to.

Group authorities are additive - a user can gain authorities by being a member of a group, but cannot lose any authorities

To create a new group use the "New Group" button.

To attach a user to a group go to the user's "User Details" tab.

Check the entry for the group that you want to add the user to in the "Groups this user belongs to" list.

[More Details about Groups](#)

Five Things You Need to Know About Basic Framework Security

These general concepts apply to the most common configurations:

User and authority data is usually stored in files VFPPF06 and VFPPF07 on the server

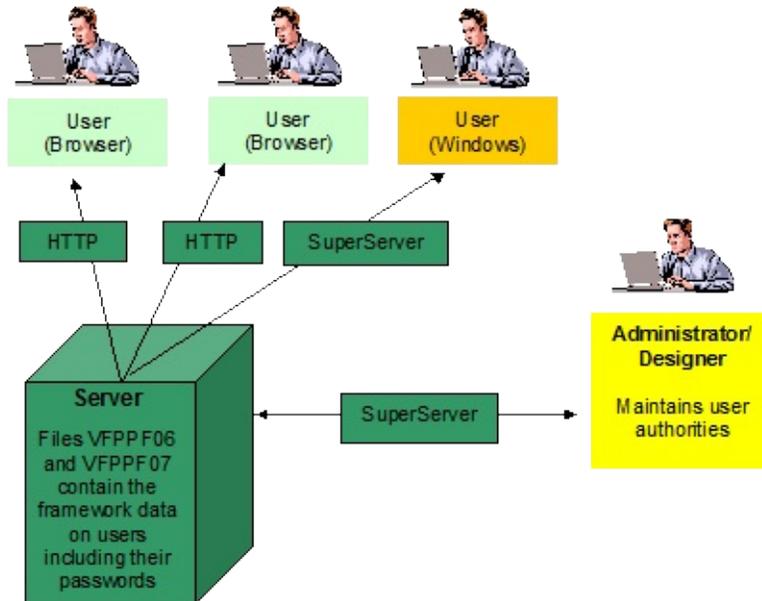
Data in VFPPF06/07 controls access to Framework objects for both Web and Windows users

There are two sets of user profiles and passwords on the server

You can assign authorities to groups of users

In design mode security is not applied

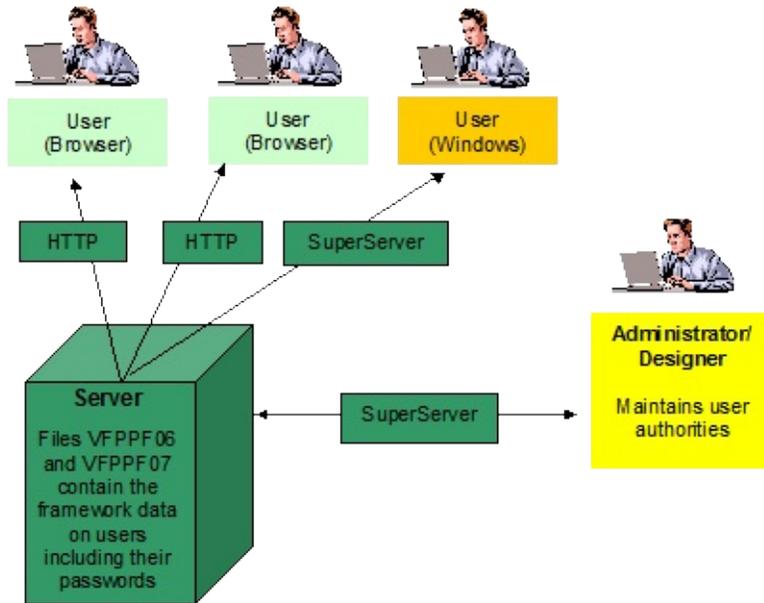
User and authority data is usually stored in files VFPPF06 and VFPPF07 on the server



The data about the users, their passwords, and their authorities to Framework objects (applications, business objects and commands) will usually be stored in files VFPPF06 and VFPPF07 in the data library of the LANSAs partition that the Framework is running in. These are ordinary LANSAs files so you can easily check whether user details have been stored on the server.

The administrator must be connected to the server at the time that the user details are saved.

Data in VFPPF06/07 controls access to Framework objects for both Web and Windows users



There are two sets of user profiles and passwords on the server

The Framework stores its own user profiles and passwords in files VFPPF06 and VFPPF07. But the server also has its own native method of storing user profiles and passwords.

When a Windows user connects to the server they use the server's native user profile and password. But once they are connected, the Framework controls which Framework objects they can access using its Framework authorities, if the user profile exists in the Framework security files.

If the Framework web sign on is switched on, the Framework will check the browser user's password against the password stored in VFPPF06/07 (entered by the administrator).

You can assign authorities to groups of users

The Framework allows you to create groups and you can assign Framework authorities to a group.

A user can be a member of several groups, or one, or none. The user gets authority to anything they are individually authorised to as well as anything that any group they belong to is authorised to.

Group authorities are additive - a user can gain authorities by being a member of a group, but cannot lose any authorities

[More Details about Groups](#)

In design mode security is not applied

When the Framework is run in design mode, Framework authority to applications / business objects / commands is not applied, because the designer needs to be able to see all Framework objects.

To test how Framework security will be applied to end-users, start the Framework using UF_EXEC, or access the Framework using a web browser.

Advanced Options

There are ways to extend or modify security to handle your site's requirements which apply to the most common configurations:

You can add your own extra layers of security checking

You can make the Framework validate user profiles and passwords using your own security system

You do not have to use Framework security

A user who signs on with one profile can be automatically changed to another

Two different Frameworks can share the same set of users

User and security data can be stored as an XML file instead of using files

VFPPF06/07

You can store other information relating to a user

You can include the user profile and password in a start URL

Export or Import of Full or Partial User Data

You can add your own extra layers of security checking

There are many overrides (Imbedded Interface Points) that you can use to make the Framework do additional checking.

These can make the Framework do your own check of:

- the choice of password (see `avPasswordRules`)
- the user profile and password (see `avPrivateConnect` (Windows) or `UFU0001` (web))
- the user's authority to an application, business object or command (see `avCheckAuth` (Windows only))
- the user's authority to instances of a business object (see `avCheckInstanceAuth` (Windows only))
- the user's authority to a particular command for a particular instance of a business object (see `avCheckInstanceAuth` (Windows only))

[Create Your Own Imbedded Interface Points \(IIP\)](#)

You can make the Framework validate user profiles and passwords using your own security system

The overrides (Imbedded Interface Points) can also be used to make the Framework validate users according to your own system.

You can have your own program validate web sign on.

You can have your own method routine do the connect to server.

(see avPrivateConnect in UF_SYSTM (Windows) or UFU0001 (Web))

See also [Create your own Imbedded Interface Points \(IIP\)](#)

You do not have to use Framework security

The Framework security is optional. If you like you can switch off Framework security and just use the server's native security (the http server for web users). In this way you can still control who accesses your application, but you won't be able to control what they can do within the Framework.

Or you can switch off the web and Windows sign on (allowing the server to handle these) but still apply the user's authorities to Framework objects.

A user who signs on with one profile can be automatically changed to another

A feature of the Imbedded Interface Points (IIPs) is that they can be used to automatically change one user profile to another. (see `avPrivateConnect` or `avSetSessionValues` (Windows) or `UFU0001` (web))

This can be useful if you have many users who fall into a few standard categories and you want to avoid registering them all in the Framework. Refer also to [More details about being signed on as a Different User](#).

Two different Frameworks can share the same set of users

If you have two Frameworks it is possible to share the same users. User details such as the profile, caption, disabled status and password are shared, but separate user authorities for each Framework are stored. Refer also to [More details about being signed on as a Different User](#).

If you want Frameworks to have entirely separate users ensure that the Frameworks have different "User Set" values.

(Go to (Framework) --> (Properties...) --> User Administration Settings tab --> User Set)

User and security data can be stored as an XML file instead of using files VFPPF06/07

A less secure option for Frameworks with no web access is to store the user definitions and authorities in local file VF_Sy001_Users.XML .

When authority is stored in VF_Sy001_Users.XML, VF_Sy001_users.XML must exist on the user's PC . This method can still be used when users connect to a remote server, as long as VF_Sy001_Users.XML is present on every user's PC, or is accessible on the network by every user's PC.

Authority stored in VF_Sy001_Users.XML must exist on the user's PC, but can be used when connecting to a server (provided it exists on every user's PC or can be accessed on the server by every user's PC).

User definitions and authorities stored in VF_SY001_Users.XML cannot be used at all when the Framework is running in web mode.

Important facts about storing user definitions as XML:

- You can change the name of the users definition file to something else.
- You can specify a complete path to the file. This could be a path to a drive on the network, so that all users could share the same XML file.
- You should never alter the content of the XML file with anything other than through the Framework.
- In the partition execute directory files named VF_Sy001_Users_YYYYMMDD_HHMMSS.XML may be found. These represent user definition versions that were saved at the time specified in their names. You can use these to back out unwanted user definition changes. You should, from time to time, delete old saved versions.
- The use of the shipped Framework user profile management facilities is optional. Not using them does not preclude your application from using multiple users and user profile management facilities of its own design.

To use this facility:

Go to (Framework) --> (Properties...) -->User Administration Settings tab and uncheck the option Store Users in DBMS Tables VFPPF067/07.

You can store other information relating to a user

The Framework will allow you to define other information that you want to store against each user.

This information can be retrieved by command handlers and filters at run time.

You could use this information to make the Framework behave differently for different users.

See [Custom Properties](#).

You can include the user profile and password in a start URL

When the user starts a Web browser VLF application, the URL used to start to web Framework can include the user profile and password

See [Web Application Start Options](#).

Export or Import of Full or Partial User Data

You can import data about users from a different system, provided that data can be accessed by a LANSAs function.

You can export user data from the Framework as an XML file (except passwords)

The imported data can replace all existing users, or it can update just some details of existing users.

The information about users that can be exported and imported includes the groups they belong to and their authorities.

You can also manually edit an exported XML file of user data and then import it.

Also see:

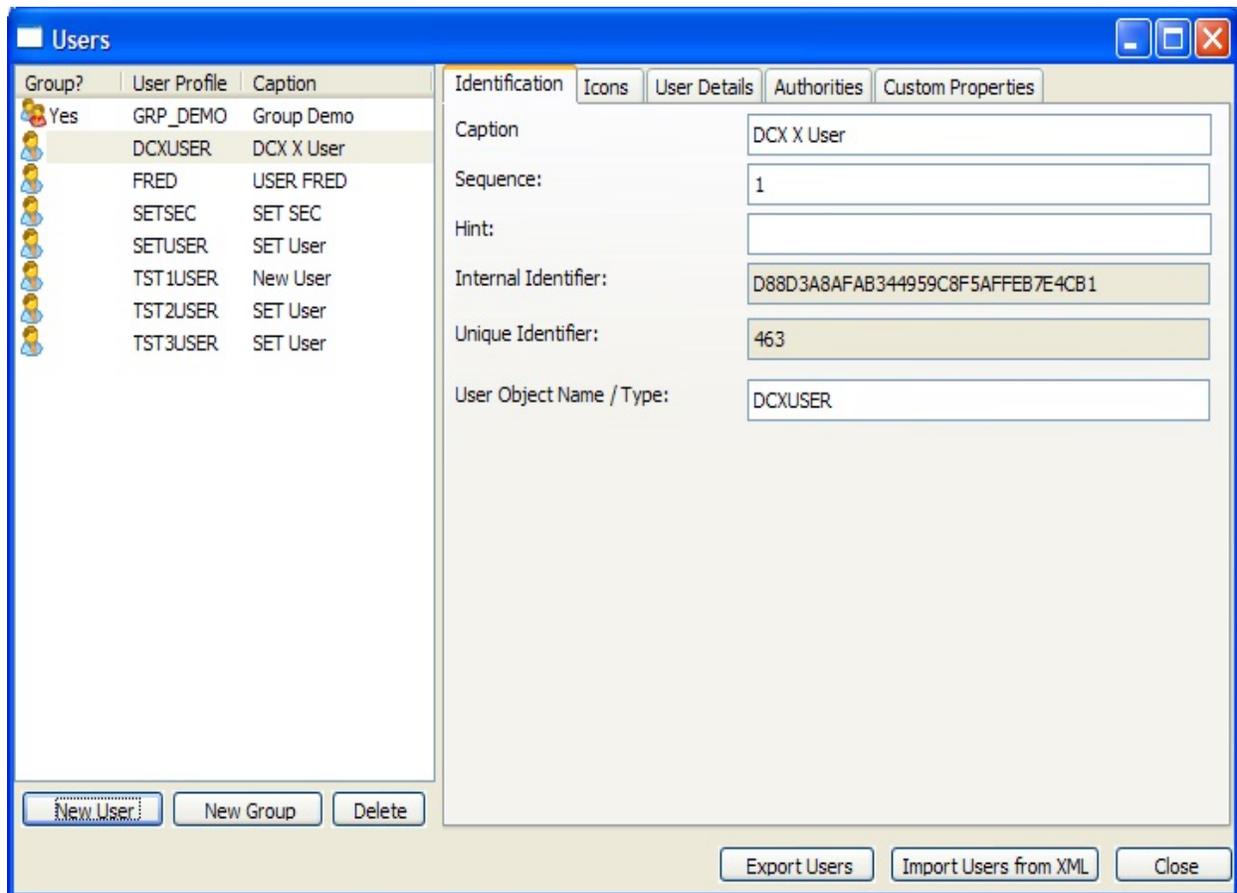
[How to Export User Data](#)

[How to Import User data from an XML file](#)

[Set up the Framework to Generate an XML file from an External Source of user data and import it](#)

How to Export User Data

Sign on as Administrator/ Designer.
(Administration) --> (Users).



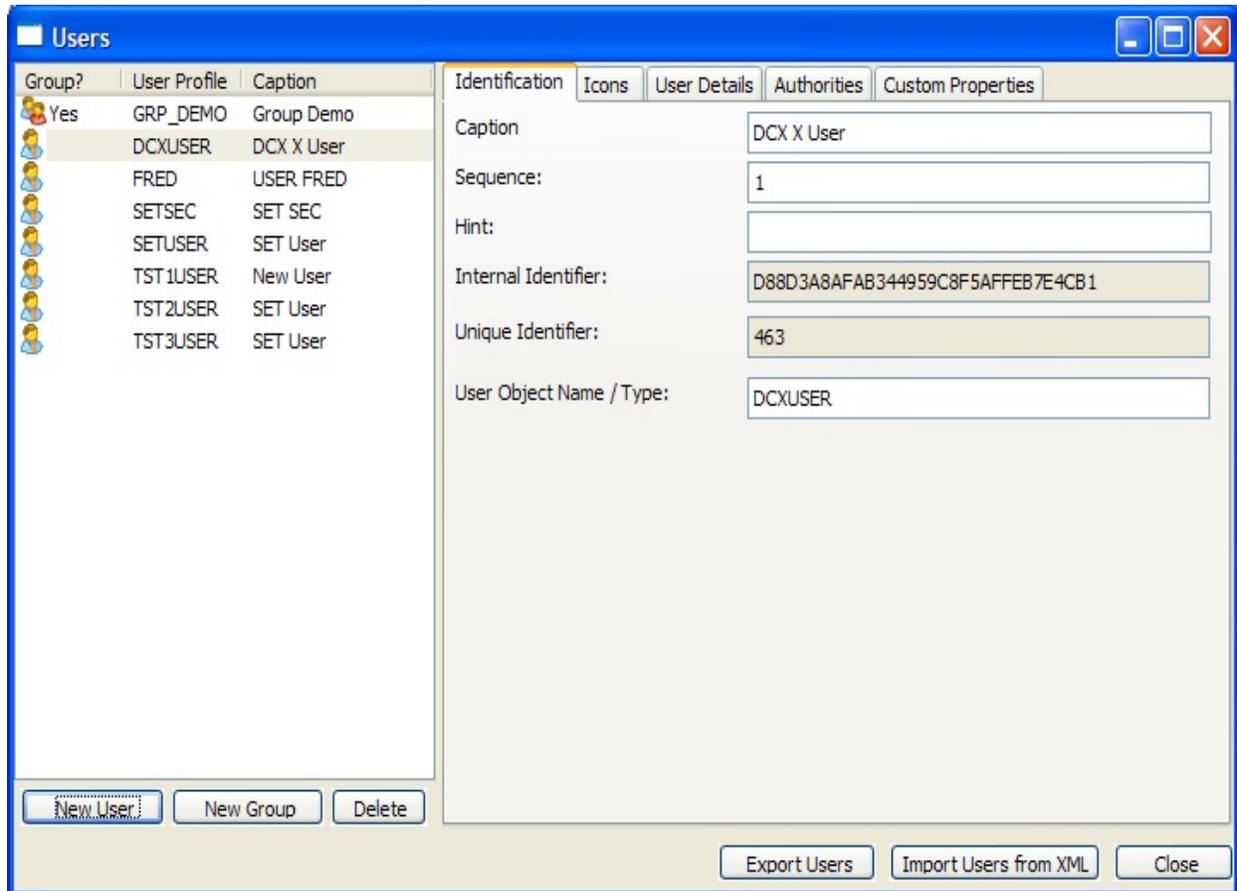
Press the *Export Users* button. The Framework will request a target directory and file name.

Press *Save* to create an XML file of User data.

How to Import User data from an XML file

Sign on as Administrator/ Designer.

(Administration) --> (Users).



Press the *Import Users from XML* button. The Framework will request a directory and filename.

Specify the name and location of your file and press *Open*.

The XML file of User data must be correctly formatted. The structure is the same as the exported XML. It should look like this:

```
<?xml version="1.0" ?>
<EXTRACT>
  <USERS ACTION="UPDATE">
    <USER ACTION="UPDATE" UUSERPROFILE="FRED">
      <USEQUENCE TYPE="N" VALUE="1" />
      <UCAPTION LANG="ENG" VALUE="USER FRED" />
      <UHINT LANG="ENG" VALUE="" />
      <UICONNAME VALUE="VF_IC496" />
    </USER>
  </USERS>
</EXTRACT>
```

```
<USEROBJECTTYPE VALUE="FRED_OBJ" />
<UPASSWORD VALUE="FREDSPSWD" />
<UEMAILADDRESS VALUE="FRED@LANSA.COM.AU" />
<UTEMPDIRECTORY VALUE="C:\DOCUME~1\user\LOCALS~1\Fred'
<UDISABLED VALUE="FALSE" />
<UADMIN VALUE="FALSE" />
<UGROUPUSER VALUE="FALSE" />
<USIGNOFFTIMEOUT TYPE="N" VALUE="0" />
<USIGNONTIMEOUT TYPE="N" VALUE="0" />
<GROUPS ACTION="REPLACE">
  <GROUP VALUE="GROUP_1" />
</GROUPS>
<AUTHORITIES ACTION="REPLACE">
  <AUTHORITY TYPE="FRAMEWORK" OBJECT="SHIPPED_FRAMI
</AUTHORITIES>
</USER>
</USERS>
</EXTRACT>
```

If you want to create a LANSa function that generates XML like this, start by looking at function UF_SYSBR/UFU0004.

Also see

[The rules for Creating an XML file of User Data](#)

Set up the Framework to Generate an XML file from an External Source of user data and import it

Read the comments in the shipped example component UF_IMPUS. (It's a Visual LANSA reusable part.)

Now create your own version of component UF_IMPUS, so that:

- a. It reads from your own source of user data.
- b. Writes it out in the standard XML format.
- c. Returns an *OK* return code and the full name of the XML file produced.

Now sign on to the Framework as Designer.

1. Select *Framework*, then *Properties* and then *User Administration Settings*.
2. Change the setting: *Import Users Imbedded Interface Point* to the name of your function.
3. Save the Framework and Exit.
4. Sign on as the Administrator or Designer
5. Select *Administration* then *Users*.

You will now see a new button available called *Import Users*.

When the Administrator presses this button, the Framework will run your function, and will then import the user data from the XML file it produces. This will create or update users in the Framework.

Users

Group? | User Profile | Caption

Yes	GRP_DEMO	Group Demo
	DCXUSER	DCX X User
	FRED	USER FRED
	SETSEC	SET SEC
	SETUSER	SET User
	TST1USER	New User
	TST2USER	SET User
	TST3USER	SET User

Identification | Icons | User Details | Authorities | Custom Properties

Caption: DCX X User

Sequence: 1

Hint:

Internal Identifier: D88D3A8AFAB344959C8F5AFFEB7E4CB1

Unique Identifier: 463

User Object Name / Type: DCXUSER

New User | New Group | Delete

Export Users | Import Users | Import Users from XML | Close

The rules for Creating an XML file of User Data

The structure, and permissible elements, are as shown in the following code example:

```
<?xml version="1.0" ?>
<EXTRACT>
  <USERS ACTION="UPDATE">
    <USER ACTION="UPDATE" UUSERPROFILE="FRED">
      <USEQUENCE TYPE="N" VALUE="1" />
      <UCAPTION LANG="ENG" VALUE="USER FRED" />
      <UHINT LANG="ENG" VALUE="" />
      <UICONNAME VALUE="VF_IC496" />
      <UUSEROBJECTTYPE VALUE="FRED_OBJ" />
      <UPASSWORD VALUE="FREDSPSWD" />
      <UEMAILADDRESS VALUE="FRED@LANSA.COM.AU" />
      <UTEMPDIRECTORY VALUE="C:\DOCUME~1\user\LOCALS~1\Fred" />
      <UDISABLED VALUE="FALSE" />
      <UADMIN VALUE="FALSE" />
      <UGROUPUSER VALUE="FALSE" />
      <USIGNOFFTIMEOUT TYPE="N" VALUE="0" />
      <USIGNONTIMEOUT TYPE="N" VALUE="0" />
      <GROUPS ACTION="REPLACE">
        <GROUP VALUE="GROUP_1" />
      </GROUPS>
      <AUTHORITIES ACTION="REPLACE">
        <AUTHORITY TYPE="FRAMEWORK" OBJECT="SHIPPED_FRAMI" />
      </AUTHORITIES>
    </USER>
  </USERS>
</EXTRACT>
```

General rules

Other element types will be ignored.

Property elements (e.g. uSequence, uCaption) have a VALUE attribute.

Numeric property elements should also have a TYPE = "N" attribute.

Boolean properties such as <UADMIN>, <UDISABLED>, <UGROUPUSER> are specified using VALUE="TRUE" or VALUE="FALSE"

The ACTION attribute can only be specified for USERS USER GROUPS or AUTHORITIES.

ACTION="UPDATE" means change/add what is specified in the XML, but leave everything else as it is. If it doesn't exist already, it will be created.

ACTION="REPLACE" means change/add what is specified in the XML and remove anything that is not specified in the XML.

ACTION="DELETE" means remove the object/s.

Rules for Specific Elements/Attributes

Element	Attribute	Description	Value/s
<USERS>	ACTION	Replace all users or add to/change existing	REPLACE
<USER>	ACTION	How to handle properties not specified in the XML, or To delete the user	REPLACE DELETE
	USERPROFILE	The User Profile of the User	
<GROUPS>	ACTION		REPLACE DELETE
<GROUP>	VALUE	The User profile of the Group. The Group should exist in the Framework already. If not, the Group should be defined as a <USER> prior to other users, in the XML.	
<AUTHORITIES>	ACTION	Replace all authorities, Add to/change existing authorities, or delete the specified authorities	REPLACE DELETE
<AUTHORITY>	TYPE	The type of object that the user will not be authorised to use.	FRAME APPLIC BUSINE COMM/ APPLIC SERVEF

OBJECT	The "User Object Name / Type" of the object the user will not be able to use.	
VALUE	Allow the user to use this object (only used with the FRAMEWORK object) or Disallow the user to use this object (all other kinds of objects)	ALLOW
COMMAND	(This only applies to Authority to COMMAND_REFERENCE objects)	
OWNER	(This only applies to Authority to COMMAND_REFERENCE objects) This is the "User Object Name / Type" of the object (Framework, application or business object) to which the command is applied	
OWNTYP	The type of object the command applies to	FRAME APPLIC BUSINE

Writing your own version of the User Authority report

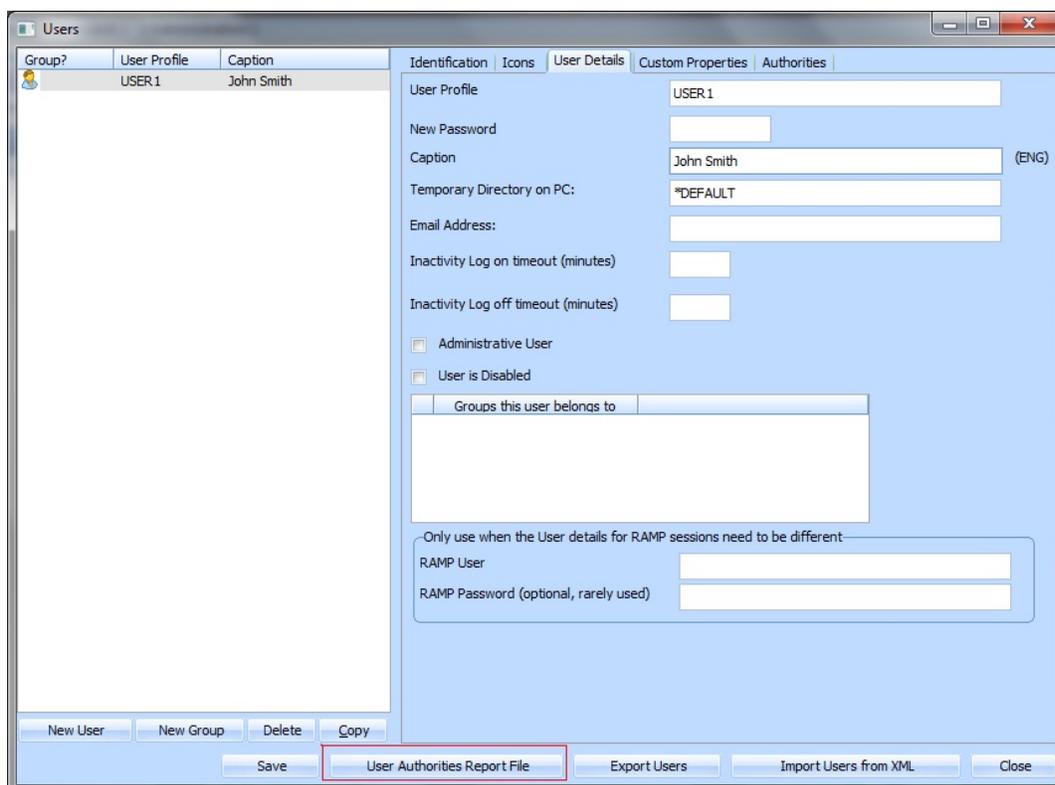
If you want a customized report to be produced when the administrator presses the User Authorities Report File button, you need to write your own report component.

To do this:

- Start by reading the comments in the shipped example component UF_REPUS. (It's a Visual LANSA reusable part.)
- Then copy UF_REPUS to make your own version of the component. First try using it unchanged so that you can see what it does. Compile it.

To test the component, sign on to the Framework as Designer. Then:

- Select Framework, then Properties and then User Administration Settings.
- Change the setting: [Report on Users - Imbedded Interface Point \(Id\)](#) to the name of your reusable part.
- Save the Framework and Exit.
- Sign on as the Administrator or Designer
- Select Administration then Users.



- Press the "User Authorities Report File" button. It will now run your component, and it should produce a .CSV file and ask you where to save it. Compare the report produced with what you want to achieve, and modify your version of the component to suit. You can request different selection criteria from the administrator, and output the data in different structures. You could also output to a database if you wish.

Frequently Asked Questions (Users and Security)

I do my unit and suite testing in Design mode. Is this a good practice?

When I run the Framework security does not seem to work?

When I start the Framework as an administrator (for example UF_ADMIN), the Framework appears briefly and then disappears?

Why options I have disabled for Windows and/or the Web browser are displayed?

Why does a user with my user profile get automatically created?

Can I control which user profile appears on the status bar?

How can my RDML programs access the user profile that the user signed on as, and the user profile shown in the status bar?

What are the user timeout settings used for?

How can I allow the user to change their own password?

I do my unit and suite testing in Design mode. Is this a good practice?

Definitely not. When working as a designer additional menu options appear, security is disabled and additional features that use more resources are enabled.

When doing unit or suite testing you should always use the UF_EXEC (user) entry point form so that you experience the same environment that a real end-user would.

When I run the Framework security does not seem to work?

Are you using the VL Framework - as Designer entry point? If you are then refer to [I do my unit and suite testing in Design mode. Is this a good practice?](#)

Why options I have disabled for Windows and/or the Web browser are displayed?

Are you using the Framework as a designer? If you are then refer to [I do my unit and suite testing in Design mode. Is this a good practice?](#)

Why does a user with my user profile get automatically created?

When you start the Framework, if you are not required to sign on and if the Framework cannot find a user with the user profile that you started LANSAs with, it will create a temporary Framework user with the appropriate user profile. It does this because it needs to have a current user.

What are the user timeout settings used for?

The timeout settings are used if you want to force the user to sign on again if they have been inactive for a certain period, or if you want the Framework to close if the user has been inactive for a certain period. Use F2 when editing the settings for more details.

Note that when the VLF is running on touch devices, the Log off Inactivity Timeout and Log on Inactivity Timeout options are disabled. You should use device specific timeout settings to control access to the device.

Can I control which user profile appears on the status bar?

Windows:

You can change the user. If you use your connect to server method (see IIP `avPrivateConnect`) you can make the user connect as one user profile and return another. Or you can modify IIP `avSetSessionValues` to give a user a different session value for user profile.

Or in client server applications, you can change the user that is shown by changing the value of field `CHK_VUSER` in the RDML of the Server IIP function to validate sign on (default is function `UFU0005`). See the source code of `UF_SYSBR/UFU0005` for more details.

Web:

You can change the user that is shown by changing the value of field `CHK_VUSER` in the RDML of the User Sign on IIP (default is function `UFU0001`).

```
CHANGE FIELD(#CHK_VUSER) TO(*User)
```

or

```
CHANGE FIELD(#CHK_VUSER) TO(*WebUser)
```

or

```
CHANGE FIELD(#CHK_VUSER) TO(#CHK_USER)
```

How can my RDML programs access the user profile that the user signed on as, and the user profile shown in the status bar?

Windows:

Change #MYUSERFIELD *USER

Web:

Change #MYUSERFIELD to *USER

or

Change #MYUSERFIELD to *WebUser

or

USE builtin(VF_GET) with_args(LOGGEDONUSER)
to_get(#MYUSERFIELD)

WAM:

Change #MYUSERFIELD #thishandler.avLoggedonUser

How can I allow the user to change their own password?

Windows:

Change the setting for Users can change their own password in the User Administration Settings tab in Framework details.

This option is only available if the option Users sign on locally to use the Framework has been selected.

Web:

There is no Framework setting to allow the user to change their own password.

A way of achieving this would be to change the User Sign on IIP (default is function UFU0001) to make the Framework check against your own user profile/password files. Then write an ordinary command handler that interfaces with your own user profile/password files and allows the user to change their password.

When I start the Framework as an administrator (for example UF_ADMIN), the Framework appears briefly and then disappears?

To use UF_ADMIN (or equivalent) the user must also be flagged as administrator.

No error message is shown in this situation because this would visibly indicate the exact cause of a security violation to a potentially insecure user.

Some Scenarios

These scenarios illustrate some of the alternatives that can be used:

Scenario: System i Web Server

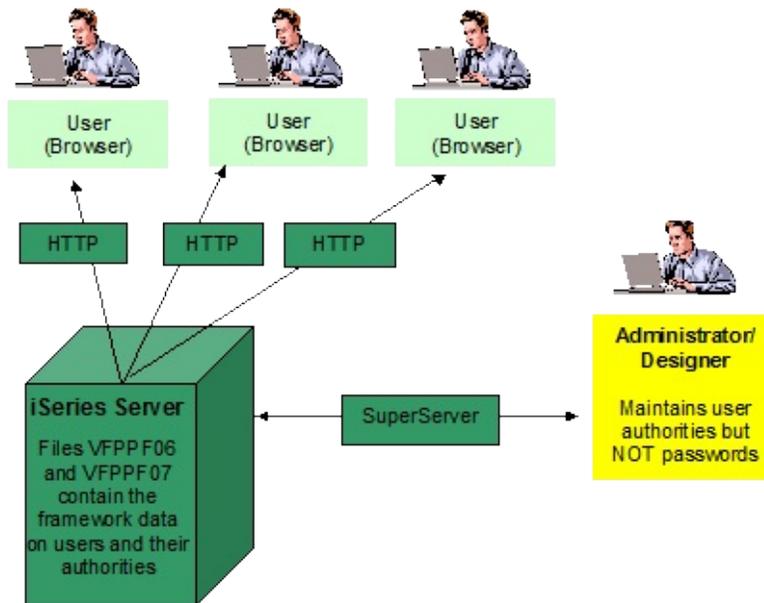
Scenario: System i Server, Users Connect with Super Server

Scenario: Windows Server, Users Connect with Super Server

Scenario: System i Web Server

Assumptions Settings Modifications to your version of UFU0001

Enroll Typical Users in the Framework



When the users connect to the System i web server, their entered password will be validated by the HTTP Server against the server's native profile/password. The IIP function UFU0001 will receive the user profile that the user logged on to the HTTP server with, and switch it to the profile of the typical user. The users will get the authorities to Framework objects that their typical user has.

Assumptions

Scenario: System i Web Settings Modifications to your version of Enroll Typical Users in the
Server UFU0001 Framework

- Users will need to maintain their own passwords (via 5250)
- The administrator will not want to maintain two sets of user profiles
- The administrator will want to control user's access to Framework objects according to group

Settings

Scenario: System i Web Assumptions Modifications to your version of Enroll Typical Users in the Server UFU0001 Framework

(Framework) --> (Properties...) --> User Administration settings tab:

Option	Setting
Use Framework authority	Checked
Store user profiles on DBMS tables VFPPF06/07	Checked
End users must sign on to this Framework	In MS-Windows Only
Users sign on to a remote server to use the Framework	Checked

Configure the HTTP server on the System i to validate user profile and password

Modifications to your version of UFU0001

[Scenario: System i Web Server](#) [Assumptions](#) [Settings](#) [Enroll Typical Users in the Framework](#)

Make a new version of UFU0001 that looks up the Framework user profile ("typical user") that typifies the user who has signed on to the HTTP server and switch to the typical user. (users who are not found can have a default typical user assigned to them).

The look up could be done using a database file of user profiles, or it could be done by calling a CL program that accesses the user's System i group profile, on the basis that the Framework's "typical users" are the same as the System i group profiles.

in the RDML in your version of UFU0001

```
CALL PGM(GETGRPPRF) PARM(#CHK_USER #GRPPRF) EXIT_USED(*N
CHANGE #CHK_USER #GRPPRF
CHANGE #CHK_VALP 'FALSE'
```

CL program GETGRPPRF could be something like:

```
PGM      PARM(&USRPRF &GRPPRF)
DCL &GRPPRF *CHAR 10
DCL &USRPRF *CHAR 10
      RTVUSRPRF USRPRF(&USRPRF) GRPPRF(&GRPPRF)
ENDPGM
```

You can also modify UFU0001 to display the user that they logged on as, rather than the "typical user" they have been changed to.

```
change #CHK_VUSER *WEBUSER
```

Compile your version of UFU0001

To change the Framework to use your version of UFU0001, go to (Framework) --> (Properties...) --> Web/RAMP Details tab. Specify your function name in the option IIP - User Sign on Function Name.

Enroll Typical Users in the Framework

[Scenario: System i Web Server](#) [Assumptions](#) [Settings](#) [Modifications to your version of UFU0001](#)

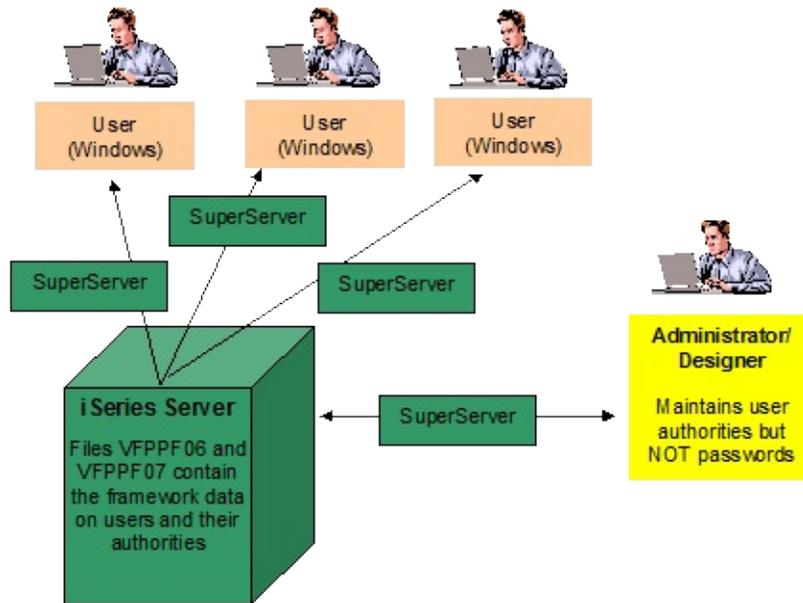
As administrator, start the Windows Framework in connect to the remote server mode.

Enroll all the typical users in the Framework. (The password does not matter)
Give the typical users authority to Framework objects that these users should have.

Save the Framework.

Scenario: System i Server, Users Connect with Super Server

[Assumptions](#) [Settings](#) [Enroll Users and Groups](#)



When the users connect to the server, their entered password will be validated against the server's native profile/password.

The users will inherit the authorities to Framework objects according to the group / groups they belong to.

Assumptions

Scenario: [System i Server, Users Connect with Super Server](#) [Settings](#) [Enroll Users and Groups](#)

- Users will need to maintain their own passwords (via 5250)
- The administrator will be able to maintain the Framework user profiles and the server user profiles
- The administrator will want to control user's access to Framework objects according to group

Settings

Scenario: System i Server, Users Connect with Super Server Assumptions Enroll Users and Groups

(Framework) --> (Properties...) --> User Administration settings tab:

Option	Setting
Use Framework authority	Checked
Store user profiles on DBMS tables VFPPF06/07	Checked
End users must sign on to this Framework	In MS-Windows Only
Users sign on to a remote server to use the Framework	Checked

Enroll Users and Groups

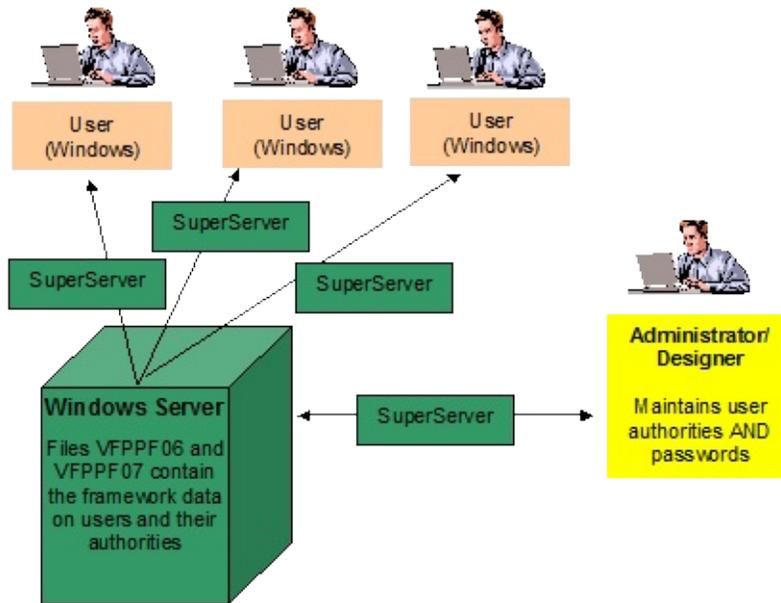
[Scenario: System i Server, Users Connect with Super Server](#) [Assumptions](#) [Settings](#)

Start the Framework in administration or design mode and:

- Create the appropriate groups
- Give each group the appropriate authorities to Framework objects
- Create the individual users (the password does not matter)
- Attach each user to the group / groups that they should belong to

Scenario: Windows Server, Users Connect with Super Server

[Assumptions](#) [Settings](#) [Enroll Users and Groups](#)



When the users connect to the server, the initial connection is made to the database using the Database Profile and Database Password (for example DBA / SQL)

Once the initial connection is made, the Framework is able to access the user profile and password information in files VFPPF06/07, and can validate the user profile and password that the user enters.

The users will inherit the authorities to Framework objects according to the group / groups they belong to.

Assumptions

Scenario: [Windows Server, Users Connect with Super Server](#) [Settings](#) [Enroll Users and Groups](#)

- Administrator will maintain users passwords
- The administrator will maintain only the Framework user profiles
- The server will be defined to the Framework with its own Database User and Database Password
- The administrator will want to control user's access to Framework objects according to group

Settings

Scenario: [Windows Server, Users Connect with Super Server](#) [Assumptions](#) [Enroll Users and Groups](#)

(Framework) --> (Properties...) --> User Administration settings tab:

Option	Setting
Use Framework authority	Checked
Store user profiles on DBMS tables VFPPF06/07	Checked
End users must sign on to this Framework	In MS-Windows Only
Users sign on to a remote server to use the Framework	Checked

(Administration) --> (Servers) --> Server Details tab:

Option	Setting
Server Type	Win
Database User	<<User profile to make the initial connection as - e.g. DBA>>
Database Password	<<password for the initial connection - e.g. SQL>>

Enroll Users and Groups

[Scenario: Windows Server, Users Connect with Super Server](#) [Assumptions](#) [Settings](#)

Start the Framework in administration or design mode and:

- Create the appropriate groups
- Give each group the appropriate authorities to Framework objects
- Create the individual users (the password does not matter)
- Attach each user to the group / groups that they should belong to

More Information

[More Details about Groups](#)

[Create Your Own Imbedded Interface Points \(IIP\)](#)

[More Details about being Signed on as a Different User](#)

[How to Share Users between Different Frameworks](#)

More Details about Groups

Creating an Authority Group

Press the New Group button. Go to the User Details tab and give the group a caption and a user profile. The group's profile does not have to exist anywhere, because groups cannot sign on.

Specifying Authorities for the Group

Go to the Authorities tab. Authorize the group to Framework objects in exactly the same way as you would a user.

Attaching a User to a Group

Click an individual user in the list on the left of the users panel and go to the User Details tab. The group that you created is shown in the list of groups at the bottom of this tab. Check the group to attach the user to it (or attach it to the user).

Authorities are Additive

If an individual user, or any of the groups they belong to, are authorized to an object, then the user can use the object.

This means that:

- If the individual user is not authorized to the object, but one of the groups they belong to is authorized, the user will be able to use the object.
- If the individual user is authorized to the object, and the user is attached to a group that is not authorized to the object, the user will still be able to use the object
- If the user belongs to two groups, and one group is authorized to an object and one group isn't, the user will be able to use the object.

Authorities of a Member of a Group

When viewing the Authorities tab, the objects to which a user has group authority are shown as shaded and protected. This is because you cannot remove users' authorities to an object to which they have group authority (because authorities are additive).

You can remove users' authorities to an object to which they have individual authority, but not group authority.

You can also authorize the individual users to an object to which they do not have group authority. This is called an individual override authority to an object.

Note about Individual Override Authority

Giving individual override authority to an object may cause the Framework to give the user individual authority to the parents of the object (if the user is not authorized to the parent already). This implementation ensures the users have a coherent individual authority even if they are removed from all groups.

The user is given individual authority only to the direct ancestors of the object. However, if new objects that belong to the same parent are subsequently added to the Framework this may result in unexpected individual authority to the new object. To avoid this problem, when adding new objects to a Framework which has already had user authorities set up, set the new object's Restricted Access property to checked. Alternatively, you can edit each user's authority after adding the new Framework object.

Custom Property Values Cannot Be Assigned to Group Profiles

Do not assign custom property values to group profiles because they will not be applied to any users belonging to the group.

Create Your Own Imbedded Interface Points (IIP)

Windows:

The Imbedded Interface Point (IIP) for Windows is the component specified in the Framework Details tab (to get there menu (Framework) --> option (Properties...) --> Framework Details tab). The option is called User Imbedded Interface Point. The shipped value for this is component UF_SYSTM.

To do your own IIP Windows checking, all you have to do is create your own version of UF_SYSTM with a different name, and then edit it and modify the relevant method routine to do what you want it to do, compile it, and then change the option from UF_SYSTM to the name of your version.

Web:

The Imbedded Interface Point (IIP) for web is the function specified in the Web/RAMP Details tab (to get there menu (Framework) --> option (Properties...) --> Web/RAMP Details tab). The option is called "IIP User Sign on function name". The shipped value for this is component UFU0001.

To do your own IIP web checking, all you have to do is create your own version of function UFU0001 with a different name, and then modify it to do what you want it to do, compile it, and then change the Framework setting from UFU0001 to the name of your version of the function.

Most of the documentation for these IIPs is in the source code of UF_SYSTM and UFU0001. Also see [What Imbedded Interface Points are Provided?](#)

More Details about being Signed on as a Different User

If you have many users who fall into a few standard categories you may want to avoid registering them all in the Framework.

Instead you can register one Framework user for each of the standard categories. Let's call these users "typical users". Give each of the typical users the correct authority to Framework objects.

Switch off the Framework web sign on option, and use the server (or HTTP server) to validate the user's password and profile.

After they have signed on, your IIP program can look up which category the user belongs to and change them to their typical user. They will then have the authorities of the typical user.

How to Share Users between Different Frameworks

In order to share users:

- The Framework must store user details on VFPPF06 and VFPPF07
- The two Framework objects must have different user object types
- The user set must be the same for both Frameworks

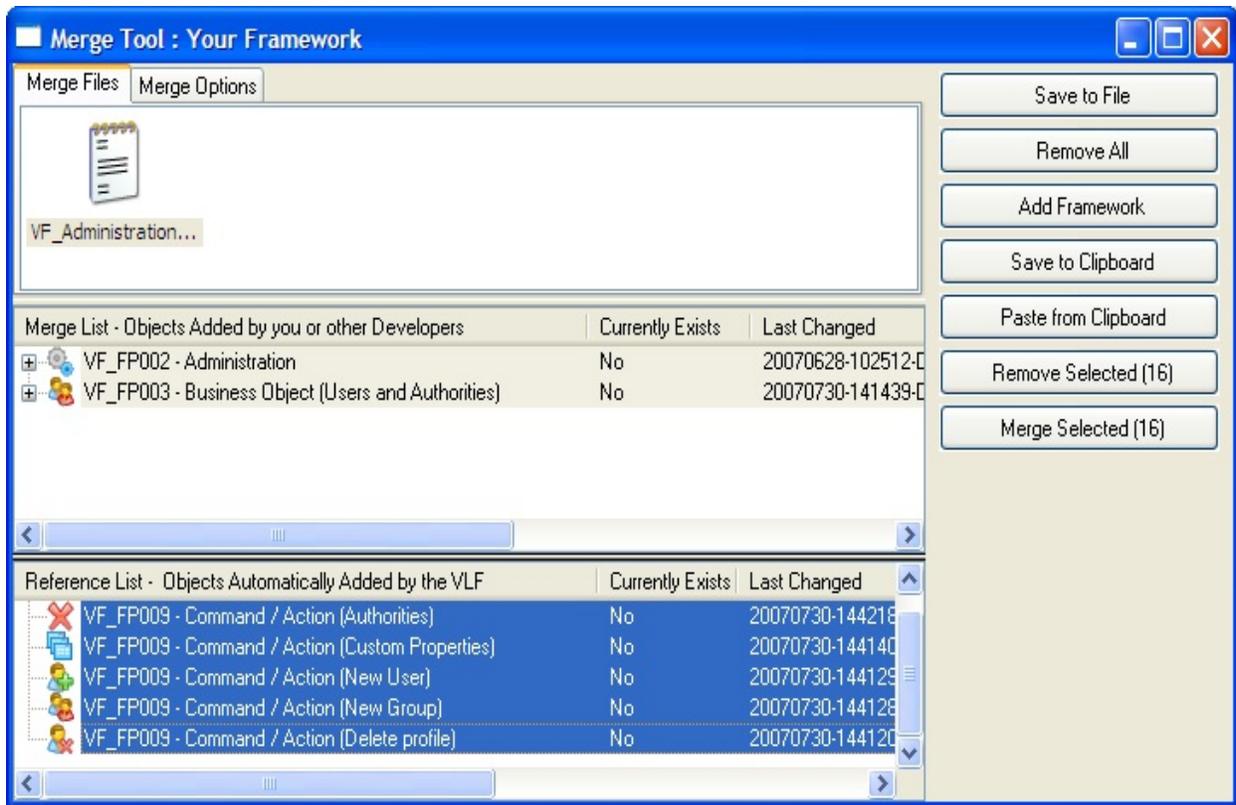
Creating Web Interface for Maintaining Users and Authorities

Note that by default the Administration application and the Users and Authorities business object are restricted access objects. You should specifically grant the Administrator authority to these objects.

It is possible to use the Administrator interface in a deployed Framework web application to create users and groups, set authorities and custom property values

To use the web interface to create and maintain users and authorities you must merge the Administration application and the Users and Authorities business object to your Framework:

1. Start the Framework in Design mode.
2. Start the Merge Tool using the Merge Tool option in the Framework menu. See [Using the Merge Tool](#).
3. Locate the merge list VF_Administration_Merge_List.txt in your partition execute folder using Windows Explorer and copy it. Paste it in the Merge Files area in the Merge Tool window.
4. To merge the Administration application and the Users and Authorities business object:
 - Select the two objects in the Merge List area in this order:
 - VF_FP002 – Administration
 - VF_FP003 – Business Object (Users and Authorities).
 - Select all the objects in the Reference List:



Note that the Merge Selected button's caption should now show Merge Selected (16).

- Click on the Merge Selected button.
- Click on the Perform All Selected Merges button.
- When you receive a message, uncheck the check box and click the Close button. **Do not Save and Restart yet.**
- Click on the Perform All Selected Merges button again.
- When you finish, **Save and Restart** the Framework but **do not upload your changes to the web server yet.**
- When the Framework restarts, make a dummy change to the Framework, save and upload your changes.

You can now administer users and authorities in a Framework web application provided you have signed on as an Administrator.

New Framework objects required for this functionality

--	--	--	--

Object type	Caption	Web Handler	Comments
Application	Administration	N/A	
Business Object	Users and Authorities	N/A	
Business Object web filter	by User Profile	Process: UF_USRM Function: UFUMF1	Is a *WEBEVENT function
Business Object commands	New User	AJAX page: UFUMUDX.htm	For all commands, use the option Is an AJAX style routine.
	New Group	AJAX page: UFUMUDX.htm	
Business Object instance commands	Details	AJAX page: UFUMUDX.htm	
	Delete Profile	AJAX page: UFUMUDX.htm	
	Custom Properties	AJAX page: UFUMUPX.htm	
	Authorities	AJAX page: UFUMUAX.htm	

Other required objects

Object type	Name	Comments	Incl depl pack
Process skeleton	UF_USRM	Standard VLF	NO

		skeleton for web functions.	
RDML Process/Function	UF_USRM/UFUMF1	Users and Authorities WEBEVENT filter	YES (follow name standard should auto include expected)
Web Pages	UFUMUDX.htm	AJAX handler for commands: New User New Group Details Delete Profile	YES AJAX (follow name standard should auto include expected target folder)
	UFUMUPX.htm	AJAX handler for Custom Properties command	
	UFUMUAX.htm	AJAX handler for User Authorities command.	
	Delete Profile	AJAX page: UFUMUDX.htm	
	Custom Properties	AJAX page: UFUMUPX.htm	
	Authorities	AJAX page: UFUMUAX.htm	

Merge list (text file)	VF_Administration_Merge_List.txt		NO
------------------------	----------------------------------	--	----

Server Profile Management and Issues

Normally server definitions are stored in a file named VF_Sy001_Servers.XML that resides in the execute directory of the associated LANSAs partition.

Some important facts about server definitions:

- You can change the name of the server's definition file to something else.
- You should never alter the content of the XML file with anything other than through the Framework.
- In the partition execute directory files named VF_Sy001_Servers_YYYYMMDD_HHMMSS.XML may be found. These represent server definition versions that were saved at the time specified in their names. You can use these to back out unwanted server definition changes. You should, from time to time, delete old saved versions.
- You do not have to use the shipped server management facilities. Not using them does not preclude your application from using multiple servers and normal LANSAs SuperServer facilities.

Deploying Server Definitions

Server definitions are changed using the UF_ADMIN entry point form or equivalent (see [Starting the Framework](#)). If you don't want this capability in your deployed applications then don't deploy UF_ADMIN or equivalent to your end-users.

The ability of end-users to edit server settings may have an effect on what you decide to ship in VF_Sy001_Servers.XML when:

- Deploying your Framework to end-user environment for the first time.
- Upgrading your Framework in end-user environments.

Some of the options you have are:

- Don't deploy VF_Sy001_Servers.XML at all. Deploy your application with instructions on how to use UF_ADMIN (or equivalent) so that end-users can define their own unique server environment.
- Centrally deploy VF_Sy001_Servers.XML with all server details predefined in it. No end-user use of UF_ADMIN (or equivalent) would be required for server definition. Normally this is only an option when only a well defined set of server systems is possible.
- Deploy VF_Sy001_Servers.XML with a predefined set of partially completed set of server examples in it. Combine this with instructions to your end-users on how to use UF_ADMIN (or equivalent) to use the shipped examples so as to define their own unique environment.

[Scenario: In House Development, Well Known Servers](#)

[Scenario: ISV, Single Unknown Server](#)

[Scenario: ISV, Multiple Unknown Servers](#)

Scenario: In House Development, Well Known Servers

Scenario: In house development applications, only allow connection to a set of well known servers.

Example: ACME engineering develops their own Framework based applications in house. They deploy them to their own PCs, which utilize one of three different System i backend servers located in different offices and manufacturing plants around the country.

Recommendation:

Ship "ready to run" so that end-users do not have to be involved in any server definition at all:

- Deploy a Framework VF_Sy001_Servers.XML.
- Deploy a LANSAC Communications Administrator LROUTE.DAT file containing the predefined server details.
- Do not allow end-users to alter the shipped details.
- Include appropriate icons into server definitions (often location based).

Scenario: ISV, Single Unknown Server

Scenario: ISV development, application only allows connection to a single (but unknown) server.

Example: ACE software develop and sell a Framework based healthcare package. They sell it to companies in the healthcare industry. It is designed to work using a single System i or Window 2000 computer as its back end server.

Recommendation:

Ship "almost ready to run" definitions so that end-user involvement in server definition administration is minimized:

- Deploy a Framework VF_Sy001_Servers.XML.
- Use a non-committal non-specific name for the server (eg: SERVER1, GLSERVER, etc).
- Deploy a LANSAs Communications Administrator LROUTE.DAT file containing the predefined server details (except for the IP address).
- End-users only need to use the LANSAs Communications Administrator to alter the "Fully Qualified Name of the Host (Address)" of the shipped definition to have the correct IP address.
- Include appropriate icon into the server definition (probably related to your Framework application).

Scenario: ISV, Multiple Unknown Servers

Scenario: ISV development, application allows connection to multiple unknown servers.

Example: XYZ software develop and sell an insurance agent package. It is designed to work using many different System i or Window 2000 computer as its back end servers. Typically the servers are located in different cities and most agents require access to two or more of the servers.

Recommendation:

Ship a set of "almost ready to run" servers so that in most situations end-user involvement is minimized. Allow end-users to define new servers in exceptional circumstances:

- Deploy a Framework VF_Sy001_Servers.XML with a predefined set of servers in it.
- Use non-committal generic names for the servers (eg: SERVER1, SERVER2, SERVER3).
- Deploy a LANSAs Communications Administrator LROUTE.DAT file containing all the predefined server details (except for the IP addresses).
- End-users should initially use UF_ADMIN to delete server definitions that are not required (leaving their definitions in LROUTE.DAT will do no harm) and to specify the caption details for the servers that are required.
- End-users then need to use the LANSAs Communications Administrator to alter the "Fully Qualified Name of the Host (Address)" of the servers that they do need to use.
- End-users may need to use the UF_ADMIN and LANSAs Communications Administrator facility to define additional servers in exceptional circumstances.

Server Connection Recovery

The Windows Framework can be configured to handle a temporary loss of connection to a server. For example, this might happen when a user's laptop moves out of range of the wireless base station.

The Framework can be configured to check the server connection:

- Prior to a move by the user to any business object or application
- Prior to executing a command
- At intervals specified by the designer.

The Framework checks the connection by attempting to run a simple function on the server.

If the reconnection attempt fails, or if the function runs but does not return an OK value, then the Framework can respond in a number of ways:

- The default response is to stop Framework activity, advise the user, and to attempt reconnect when the user clicks ok.
- The application designer also has the option of writing their own server connection test function. This could be coded to return a connection error code, even if the function runs ok. This might be useful for advising all Framework users that an upgrade is in progress and they need to log off. (See example function UF_SYSBR/UFU0004)
- Programmatic connection checking in Framework filters and command handlers is also available, using the `#avFrameworkManager.avCheckConnection` method. This would need to be coded into filters just prior to doing a database search, or coded into command handlers just prior to doing a save.
- Filters and command handlers can also detect that connection recovery is underway by listening for events signaled by the Framework manager such as `avSessRecoverStarted`, `avSessRecovered` and `avSessRecoverFailed`.

Some things worth knowing

A server connection check will not happen while your RDML(X) code is executing unless the code specifically requests `avCheckConnection`.

Even if the "Check before executing command" option is checked, you will still need to include `invoke #avFrameworkManager.avCheckConnection` in your command handlers and filters prior to doing any server database IO. (for

example, prior to doing searches and saves).

To use connection recovery you need to make sure any database commitment control boundaries you use do not span user interactions. A database commitment control boundary cannot span a session reconnection.

If you are using connection recovery you should avoid using session based values (eg: the server *JOBNBR) as 'keys' to information in your application session because they will probably change after a reconnection.

The session recovery feature deals with LANSA super-sever sessions only. It does not recover a 5250 RAMP session.

If the Framework is locked by Framework locking, connection checking and recovery is not performed, except for these situations:

- Automatic connection checks that occur at timed intervals
- Specific filter and command handler usage of avCheckConnection

Server Connection Recovery Settings

These settings are available in the Server properties tab:

[Attempt Automatic session recovery](#)

[Time interval between checks of connection status](#)

[Check connection before executing commands](#)

[Check connection before selecting applications and business objects](#)

[Action to take when session cannot be recovered](#)

[Check Connection using function](#)

Attempt Automatic session recovery

Switches on server connection checking by the Framework.

When it is switched off, you can still code `#avFramework.avCheckConnection` in your command handlers and filters

The default is unchecked.

This property is in the [Server Details](#) tab.

Time interval between checks of connection status

This specifies the interval between periodic checks that the server is still connected. 0 means the checks are disabled.

If a server connection check occurs for some other reason (eg: filter or command handler executes the `avCheckConnection` method) the timer is reset, so a full time interval elapses before the next check is performed.

If the user cancels a request to reattempt connection, the periodic checks are stopped.

If after an unsuccessful reconnection attempt, the "Action to take when the session cannot be recovered" is ABORT, the periodic checks are stopped.

The default is 30 seconds.

This property is in the [Server Details](#) tab.

Check connection before executing commands

When checked, the Framework will check that the server connection is ok, just prior to executing commands. If the connection cannot be recovered, the command will not be executed.

The default is checked.

This property is in the [Server Details](#) tab.

Check connection before selecting applications and business objects

When checked, the Framework will check that the server connection is ok, just prior to moving to a new business object or application. If the connection cannot be re-established, the move will not occur. This includes programmatic switching.

The default is checked.

This property is in the [Server Details](#) tab.

Action to take when session cannot be recovered

Retry/Notify - If the first reconnect attempt fails, notify the user and when they press OK attempt to reconnect. Continue to loop until a successful reconnect occurs, or the user presses Cancel. If the user presses cancel, no more checks will be made until the user attempts an action that results in a check connection.

Abort - If the first reconnect attempt fails, notify the user, and then stop. No more checks will be made until the user attempts an action that results in a check connection.

This property is in the [Server Details](#) tab.

Check Connection using function

This is the name of the function that is run on the server to determine whether the super server connection is active or not. See UF_SYSBR/UFU0004 for details.

Refer to the documentation in the source code of the shipped function UF_SYSBR/UFU0004 for details of how to make your own version and the conventions it should follow.

This property is in the [Server Details](#) tab.

Programmatic server connection checking

#avFrameworkManager.avCheckConnection method is available in filter and command handler code.

It should be used to check the server connection within a command handler or filter just prior to performing an operation that will use the connection (for example a search or save).

Invoking this method causes the server check function to be called on the server, so it is advisable to consider, and verify by testing, the performance ramifications of how often you invoke it. For example, invoking it in every iteration of a loop that inserts 100 database rows would be ill advised.

Method: avCheckConnection

Parameters:

Name	Usage	Class	Description	Default
Server	Input - Optional	Alpha – max length 32	The server to check (user object name/type of the server)	Current Server
AttemptReconnect	Input – Optional	Boolean	If the connection is found to have failed, attempt one reconnect	TRUE
IssueMessages	Input – Optional	Boolean	Issue messages advising the user that the connection has failed If this is set to FALSE, the ActionOnFail parameter is ignored, and no attempt is made to communicate with the user.	TRUE
ActionOnFail	Input – Optional	Boolean	Action if the one attempt to reconnect fails. Can be ABORT or NOTIFY/RETRY <ul style="list-style-type: none">• Abort	The serve value (set in the server properties

- the user is advised, no further action is taken, the timer is deactivated.
- If the user subsequently initiates `avCheckConnection`, another single attempt to reconnect is made.
- Notify/Retry
 - notifies the user, waits for their instruction to try again, and then tries again, if they press OK.
 - On each loop the user can chose OK or Cancel. If they choose Cancel:
 - No further action is taken, the timer is deactivated.
 - If the user subsequently initiates `avCheckConnection`, the whole notify/retry cycle starts again.

OfflineOK

Input -
Optional

Boolean

If this is TRUE and the user has never connected to this server (e.g. they are

TRUE

working offline, or the system has been defined to use the local database), avCheckConnection will return OK.

If this is FALSE and the user has never connected to this server, avCheckConnection will return ER

Return Value

Output - Alpha – Normally OK (now
Optional max connected) or ER (not
length 2 connected). But the
connection testing function
IIP can be coded to pass
back other values

Examples

Examples

```
EVTROUTINE HANDLING(#PHBN_1.Click)
* basic
invoke #avFrameworkManager.avCheckConnection ReturnValue(#df_elretc)
if '#df_elretc *eq OK'
invoke #Com_Owner.uSelectData
endif
ENDROUTINE
```

```
EVTROUTINE HANDLING(#PHBN_2.Click)
* silent (Attempt one reconnect if unconnected, don't advise the user of
anything - this program will advise)
invoke #avFrameworkManager.avCheckConnection
AttemptReconnect(TRUE) IssueMessages(FALSE) ActionOnFail(ABORT)
ReturnValue(#df_elretc)
if '#df_elretc *eq OK'
invoke #Com_Owner.uSelectData
else
USE BUILTIN(Message_box_show) WITH_ARGS(*Default *Default
*Default *Default 'My own message advising that connection is disabled')
endif
ENDROUTINE
```

```
EVTROUTINE HANDLING(#PHBN_3.Click)
* detect without even one attempt to recover - ActionOnFail will be irrelevant
* this will only detect that the super server connection has been lost, even if
the TCPIP link has subsequently been restored
* (to detect that you are able to connect requires an attempt to reconnect)
invoke #avFrameworkManager.avCheckConnection
AttemptReconnect(FALSE) IssueMessages(TRUE) ActionOnFail(ABORT)
ReturnValue(#df_elretc)
if '#df_elretc *eq OK'
invoke #Com_Owner.uSelectData
endif
ENDROUTINE
```

```
EVTROUTINE HANDLING(#PHBN_4.Click)
* attempt one reconnection, but if that fails, advise the user but do not retry
invoke #avFrameworkManager.avCheckConnection
AttemptReconnect(TRUE) IssueMessages(TRUE) ActionOnFail(ABORT)
ReturnValue(#df_elretc)
if '#df_elretc *eq OK'
invoke #Com_Owner.uSelectData
endif
ENDROUTINE
```

Public Signals

The following events can be listened for in command handlers and filters

`avSessRecoverStarted` - This indicates that the Framework has detected that the server connection has been lost and a session recovery is being started.

`avSessRecovered` - This indicates that the Framework has successfully restored the connection, by connecting to the server again

`avSessRecoverFailed` - This indicates that reconnection attempts have failed and have been abandoned for now

Examples

```
EVTROUTINE #avFrameworkManager.avSessRecoverstarted

set #SEARCH_BUTTON enabled(false)
invoke #avFrameworkManager.avRecordTrace component(#com_Owner)
event('Filter heard avSessRecoverStarted')

ENDROUTINE

EVTROUTINE #avFrameworkManager.avSessRecovered

set #SEARCH_BUTTON enabled(true)
invoke #avFrameworkManager.avRecordTrace component(#com_Owner)
event('Filter heard avSessRecovered Sucessfully')

ENDROUTINE

EVTROUTINE #avFrameworkManager.avSessRecoverFailed

invoke #avFrameworkManager.avRecordTrace component(#com_Owner)
event('Filter heard avSessRecoverFailed')
set #SEARCH_BUTTON enabled(false)

ENDROUTINE
```

Multilingual Application Issues

When building filters and command handlers for use in multilingual environments use exactly the same procedures as you would for any other Visual LANSA application.

The only difference between designing any multilingual Visual LANSA application and a Framework based Visual LANSA application is in the input of the multilingual Framework design details.

This input process mostly involves the captions used for the Framework, the applications, the business objects, the commands and any "Help About" details.

Specifying multilingual details for the Framework details is quite simple.

Assume that you have a Framework that was designed using language code ENG (English). You now want to translate the Framework details to language code FRA (French).

You would:

1. Invoke the UF_DESGN (or equivalent) entry point for the Framework. Make sure that the language code specified is the language that you are interested in translating to (eg: FRA for French).
2. Display the various Framework, application, business object and command property dialogs and simply overtype the displayed captions with the translated captions (initially you should do this for just a few things and verify that they work correctly (see step 3) before completing the translation job).
3. Close down the Framework. The changed details will be saved. You should now find that if you execute the Framework with language code ENG you get English details displayed, if you execute the Framework with language code FRA you should get the French details displayed.

Technically, there are two things about the way translated strings are processed you may need to understand:

1. The Framework definition file `vf_sy001_System.xml` (or equivalent) holds all translated strings for all languages. If you look in this file you may find XMI

example (which is the definition of the EXIT command):

```
<MEMBER TYPE="COMMAND" CLASS="VF_FP009" ID="COMML
  <PROPERTY NAME="UBITMAPNAME" VALUE="VF_BM013"/>
  <PROPERTY NAME="UCAPTION" LANG="ENG" VALUE="Exit"/
  <PROPERTY NAME="UICONNAME" VALUE="VF_IC035"/>
  <PROPERTY NAME="USEQUENCE" TYPE="N" VALUE="2"/>
</MEMBER>
```

you can see the definition of the caption property:

```
<PROPERTY NAME="UCAPTION" LANG="ENG" VALUE="Exit"/
```

After translation using the methods previously described (into French and Italian) then they would now appear in the XML file as:

```
<PROPERTY NAME="UCAPTION" LANG="ENG" VALUE="Exit"/
<PROPERTY NAME="UCAPTION" LANG="FRA" VALUE="Sorter'
<PROPERTY NAME="UCAPTION" LANG="ITL" VALUE="Uscire"
```

2. When language dependent properties from the Framework definition file `vf_sy001_System.xml` are being processed, the logic used to locate them goes like this:

- Look for property in the current language
- If not found, and current language is not English, try English
- If not found, and current language is English, try "NAT" (National language)

Imbedded Interface Points (IIPs)

The Framework is shipped with a large number of imbedded interface points (IIPs).

IIPs are places at which externally exposed code is invoked to perform specific internal (or imbedded) logic while the Framework is executing.

For example, in Windows applications there is an IIP method named `avConnectFiles` that defines how files are to be connected up to a server system by the Framework.

This standard shipped IIP version does this:

```
Mthroutine avConnectFiles options(*Redefine)
* ==> Define_map *input #std_obj #UserProfile
* ==> Define_map *input #vf_elenum #DftBlockSize
* ==> Define_map *input #vf_elenum #DftMaxRecSel

USE BUILTIN(CONNECT_FILE) WITH_ARGS('*' *SSERVER_SSN #DftB
                                     #DftMaxRecSel.Value)

Endroutine
```

If you want you can modify this shipped IIP logic to do something different for your Framework.

In Windows applications the IIPs are defined as methods in the shipped component `UF_SYSTM`.

In Web browser applications the IIPs are defined as RDML functions in the shipped process `UF_SYSBR`.

If you want to learn more about IIPs then a good place to start is by looking at the source code shipped in component `UF_SYSTM` (for Windows applications) and in the functions contained in the process `UF_SYSBR` (for Web browser applications).

What Imbedded Interface Points are Provided?

Windows

Web

Both Windows and Web

Windows

In **Windows** Frameworks these IIPs are available:

Method	Description
avCheckAuth Method	This method allows you to add your own layer of security on top the Framework security.
avCheckInstanceAuth Method	This method allows you to add instance-level security on top the Framework security.
AvCheckUserLicense	This IIP method allows you to license your Framework.
avCloseMAINWindow	Main window is to be closed
avCloseSECONDWindow	Secondary window is to be closed
AvConnectFiles	Connects file to a server system.
AvDisConnectFiles	Disconnects files from a server system.
AvEnrollVisualStyles	This method allows you to enroll your own visual styles.
avMAINWindowReady	Main window has opened and is ready for work
AvPasswordRules	This IIP method allows the Framework designer to code rules that new passwords must obey. These rules only apply to passwords created for local sign on. Examples of such rules might be "Must be more than 6 characters" or "Must not contain the same letter twice" or "Must contain at least one digit".
AvPrivateConnect	This IIP method replaces the standard Framework logon / connection process with a private version.
AvPrivateDisconnect	This IIP method replaces the standard Framework disconnection with a private version.

avQFActionSelection	This controls what happens when the clicks on one of the entries located in the quick find dialog. (If Quick Find Override Feature is activated).
avQFLoadSearchList	This overrides the list of possible values that can be searched for, in the Quick Find dialog. (If Quick Find Override Feature is activated).
avSECONDWindowReady	Secondary window has opened and is ready for work
avSetBCSessionValues	Sets session values just before the connection to the server occurs. Most session values are set by IIP avSetSessionValues, but since this occurs after connection to the server, it is not appropriate for some session values (for example session value PSRR). Use this IIP to set the session values that need to be set prior to connection.
AvSetSessionValues	Sets session values (via Built-In SET_SESSION_VALUE) for users at sign on time.
avValidateLANSAName	This IIP method is used to check the names of functions, processes, wams and reusable parts generated by the Program Coding Assistant
AvValidateUser	Supports imbedded validation of user profiles and passwords.
UF_SYSBR/UFU0005	This server function allows Windows Client Server applications to modify the user profile after the user has connected to the server. This allows framework security to be based on a different user than the user profile that was used to connect to the server. A typical use of this function would be to create a VLF user profile for each category of user, and assign the appropriate VLF

	<p>authorities to this user profile. Then when individual users sign on to the server, this function allocates them to the appropriate VLF user based on user data held on the server, or based on their iSeries group profile.</p> <p>See the source code of UF_SYSBR/UFU0005 for more details</p>
--	---

RAMP Dynamic Naming IIPs

avMakeControlName	This method is called each time a cell in the Input/Output control grid in Dynamic Naming interface receives the focus. It allows you to standardise the screen's field names should your application use a certain naming convention.
avMakeFormName	This method is called when a form has yet to be named using Dynamic Naming.
avValidateControlNam	Validate the name given to a Newlook screen's control.
avValidateFormName	Validate the name given to a Newlook screen.

Refer to the shipped component UF_SYSTM for more details of these methods and how to change them.

Web

In **Web browser** Frameworks these IIPs are available:

RDML Function	Description
UF_SYSBR/UFU0001	Validates the user profile and password of a user accessing a web-based Framework.
UF_SYSWB/UFU0002	Displays field and function level help

Refer to the shipped processes UF_SYSBR and UF_SYSWB for more details of these functions and how to change them.

Both Windows and Web

Both Web browser and Windows Frameworks may use these IIPs:

RDML Function	Description
UF_SYSBR/UFU0003	This method allows you to use your own multilingual text for Framework captions such as the "Clear List" on the Instance list. See MTXT String Loader .
UF_SYSBR/UFU0006	Use to allow end-users to change the IBM i password in the Framework logon dialog. Typically you would change this function to add your site's specific password rules.
UF_SYSBR/UFU0010	Default Code Table Data handler. This function stores and retrieves data for code tables that do not have their own Code Table Data handler.
UF_SYSBR/UFU0011	Code Table Data handler demonstrating that data can be stored and retrieved from file DEPTAB, for Code Table VF_DEPTAB.
UF_SYSBR/UFU0012	Code Table Data handler demonstrating that data can be retrieved from hard coded values in the function, for Code Table VF_SEX.
UF_SYSBR/UFU0013	Code Table Data handler demonstrating that data can be retrieved from a .dat file on a PC, or from hard coded values when running on the System i, for Code Table VF_COUNTRY.
UF_SYSBR/UFU0014	Code Table Data handler demonstrating that data can be retrieved from a .dat file on a PC, or from hard coded values when running on the System i, for Code Table VF_CURRENCY.
UF_SYSBR/UFU0015	Code Table Data handler demonstrating that data can be retrieved from a .dat file on a PC, or from hard coded values when running on the System i, for Code Table VF_USASTATES.

Refer to the shipped process UF_SYSBR for more details of these functions and how to change them.

Adding your own object security validation

In Windows applications you can add your own layer of object security on top of the Framework security by modifying the IIP method `avCheckAuth`. This can be used to make the Framework security conform to your site's standards. It can also be used to alter the behaviour of filters or command handlers for particular users.

The IIP routine receives an identifier for the object (This is the user object name / type parameter which can be viewed on the identification tab sheet for the object) and the user profile.

A special case is when the object being validated is a command reference (a command for a particular Framework or application or business object) the IIP receives the user profile, an identifier for the Framework/application/business object, and an identifier for the command. (The identifier is the user object name / type parameter).

See the source of `UF_SYSTM`, method `avCheckAuth`, for details.

The IIP routine could be changed to do something like `FETCH` a security record for this user and object from a file containing authorization data.

The IIP routine returns a result as `OK` or `ER`. When the Framework is run in a mode other than design, the Framework will evaluate the IIP routine for each object, and automatically display it or hide it as appropriate.

The security of the IIP routine is additive with the Framework security - If the Framework's security says that the user is not authorized to the object, OR the IIP routine says that the user is not authorized to the object, the object won't be displayed for that user.

If you want to, you can access `avCheckAuth` from the code in your filters and command handlers, and alter their behaviour based on the returned "additional information parameter".

When you invoke `avCheckAuth` from a filter you are validating the business object. For the business object to have been visible in the first place, the IIP routine must have returned `OK` for that business object, but it could also return the additional information parameter. The filter could use this to decide whether to restrict filter options for particular users.

When you invoke `avCheckAuth` from a command handler you are validating a command reference (a Framework/application/business object - command combination). For the command to be available for the

Framework/application/business object, the IIP routine must have returned OK for this command reference, but it could also return the additional information parameter. The command handler could use this parameter to allow some users to edit all fields while restricting other users to work with a reduced set of fields, or to view only access.

One point to remember when writing an avCheckAuth routine is that if your users connect to a server, they will not be connected to the server at the time their authority to the Framework or servers are evaluated. However they will be connected at the time their authority to applications, business objects and commands are evaluated.

avCheckAuth Method

avCheckAuth Method (when invoked from Command Handlers and Filters).

Method: avCheckAuth (when invoked from a filter)

Parameters:

Name	Usage	Class	Description
ReturnCode	Output - Mandatory	Alpha - max length 5	According to the IIP avCheckAuth: OK - the user is authorized to this business object ER - the user is not authorized to this business object
UserAuthInformation	Both-Optional	Alpha - max length 75	This is optional additional information about the current user and the current business object, that can be passed from the IIP avCheckAuth in UF_SYSTM back to the filter.

Method: avCheckAuth (when invoked from a command handler)

Parameters:

Name	Usage	Class	Description
ReturnCode	Output - Mandatory	Alpha - max length 5	According to the IIP avCheckAuth: OK - the user is authorized to this command reference ER - the user is not authorized to this command reference
UserAuthInformation	Output - Optional	Alpha - max length 75	This is optional additional information about the current user, the current business object, and the current command, that

		can be passed from the IIP avCheckAuth back to the filter.
--	--	---

Adding your own instance level security validation

You can add instance level security to the Framework by modifying the IIP `avCheckInstanceAuth`. For example if we take the employees business object in the demonstration system, the same user can be authorized to different commands depending on the employee selected. This can be used to strengthen security or to make the Framework security conform to your site's standards. It can also be used to alter the behaviour of filters or command handlers for particular users and particular business object instances.

The IIP routine receives an identifier for the business object and an identifier for the command (In both cases the identifier is the user object name / type parameter which can be viewed on the identification tab sheet for the business object or command), the user profile, and the identifying keys for the business object instance.

(A special case is if `avCheckInstanceAuth` is invoked by a filter. In this case no identifier for the command is specified).

See the source of `UF_SYSTM`, method `avCheckInstanceAuth`, for details.

The IIP routine could be changed to do something like `FETCH` a security record for this user profile, business object, command, and business object instance. Based on this security record it can return `OK` or `ER`.

The IIP routine returns a result as `OK` or `ER`. When the Framework is run in a mode other than design, the Framework will evaluate the IIP routine for each business object instance, and automatically display or hide the commands appropriate to that business object instance and user.

The security of the `avCheckInstanceAuth` IIP routine is additive with the Framework security - If the Framework's security says that the user is not authorized to the object (application/business object/command reference), OR the IIP routine says that the user is not authorized to the command for that instance of the business object, the command won't be displayed for that user.

If you want to, you can access `avCheckInstanceAuth` from the code in your filters and command handlers, and alter their behaviour based on the returned "additional information parameter".

When you invoke `avCheckInstanceAuth` from a filter you are validating the instance of the business object (for any command). Based on the return code and the additional information parameter the filter could decide (for example) which of the data selected from the database should get added to the instance list.

When you invoke `avCheckInstanceAuth` from a command handler you are validating a command reference (a Framework/application/business object - command combination) for a particular business object instance. For the command to be available for this business object instance, the IIP routine must have returned OK for this command reference and business object instance, but it could also return the additional information parameter. The command handler could use this parameter to allow (instance by instance) some users to edit all fields while restricting other users to work with a reduced set of fields, or view only access.

avCheckInstanceAuth Method

avCheckInstanceAuth Method (when invoked from Command Handlers and Filters).

Method: avCheckInstanceAuth (when invoked from a filter)

Parameters:

Name	Usage	Class	Description
ReturnCode	Output - Mandatory	Alpha - max length 5	According to the IIP avCheckInstanceAuth: OK - the user is authorized to this business object instance ER - the user is not authorized to this business object instance
UserAuthInformation	Output - Optional	Alpha - max length 75	This is optional additional information about the current user and the current business object instance that can be passed from the IIP avCheckInstanceAuth back to the filter.
AKey1 AKey2 AKey3 AKey4 AKey5	Input - Optional	Alpha - max length 32	These are the optional alphanumeric programmatic identifiers of this business object instance. The identification protocol used for the identifier is at your discretion.
NKey1 NKey2 NKey3 NKey4 NKey5	Input - Optional	Numeric - max (15,0) precision	These are the optional numeric programmatic identifier of this business object instance. The identification protocol used for the identifier is at your discretion.

Method: avCheckInstanceAuth (when invoked from a command handler)

Parameters:

Name	Usage	Class	Description
ReturnCode	Output - Mandatory	Alpha - max length 5	According to the IIP avCheckInstanceAuth: OK - the user is authorized to this command for this business object instance ER - the user is not authorized to this command for this business object instance
UserAuthInformation	Output - Optional	Alpha - max length 75	This is optional additional information about the current user, the current business object instance and the current command that can be passed from the IIP avCheckInstanceAuth back to the filter.
AKey1 AKey2 AKey3 AKey4 AKey5	Input - Optional	Alpha - max length 32	These are the optional alphanumeric programmatic identifiers of this business object instance. The identification protocol used for the identifier is at your discretion.
NKey1 NKey2 NKey3 NKey4 NKey5	Input - Optional	Numeric - max (15,0) precision	These are the optional numeric programmatic identifier of this business object instance. The identification protocol used for the identifier is at your

			discretion.
--	--	--	-------------

avCheckInstanceAuth Example

Example of an avCheckInstanceAuth User IIP in UF_SYSTM (or equivalent):

```
Mthroutine Name(avCheckInstanceAuth) Options(*Redefine)
* ==> Define_Map For(*input) Class(#std_obj) Name(#UserProfile)
* ==> Define_Map for(*input) class(#vf_eltxts) Name(#ObjectType)
* ==> Define_Map For(*input) Class(#vf_elidn) Name(#BusObjectName)
* ==> Define_Map For(*input) Class(#vf_elidn) Name(#CmdObjectName) m
* ==> Define_Map For(*output) Class(#std_Bool) Name(#ReturnCode)
* ==> Define_Map for(*output) class(#std_TextL) Name(#UserAuthInformati

* ==> Define_map for(*input) class(#vf_elXAK1) Name(#AKey1)
* ==> Define_map for(*input) class(#vf_elXAK2) Name(#AKey2)
* ==> Define_map for(*input) class(#vf_elXAK3) Name(#AKey3)
* ==> Define_map for(*input) class(#vf_elXAK4) Name(#AKey4)
* ==> Define_map for(*input) class(#vf_elXAK5) Name(#AKey5)
* ==> Define_map for(*input) class(#vf_elXNK1) Name(#NKey1)
* ==> Define_map for(*input) class(#vf_elXNK2) Name(#NKey2)
* ==> Define_map for(*input) class(#vf_elXNK3) Name(#NKey3)
* ==> Define_map for(*input) class(#vf_elXNK4) Name(#NKey4)
* ==> Define_map for(*input) class(#vf_elXNK5) Name(#NKey5)

Set Com(#ReturnCode) Value(OK)

Define Field(#UF_WKOBJT) Reffld(#VF_ELIDN)
Define Field(#UF_WKOBJ1) Reffld(#VF_ELIDN)
Define Field(#UF_WKOBJ2) Reffld(#VF_ELIDN)

Change Field(#UF_WKOBJT) To('#OBJECTTYPE.VALUE')
Change Field(#UF_WKOBJ1) To('#BusObjectName.VALUE')
Change Field(#UF_WKOBJ2) To('#CmdObjectName.VALUE')

Case Of_Field(#UF_WKOBJT)
When Value_Is('= BUSINESS_OBJECT')

Case Of_Field(#UF_WKOBJ1)
When Value_Is('= EMPLOYEES')
```

```
Case Of_Field(#UF_WKOBJ2)
When Value_Is('= SKILLS')
* invoked by the Framework to determine
* whether to show a command tab
If Cond('#AKEY1.VALUE *eq A1002')
Set Com(#ReturnCode) Value(ER)
Endif
When Value_Is('= VIDEO')
* invoked by the Framework to determine
* whether to show a command tab
If Cond('#AKEY1.Value *eq A1001')
Set Com(#ReturnCode) Value(ER)
Endif
when '= *blanks'
* this routine has been invoked from
* a filter in order to determine whether
* to add an entry to the instance list
If Cond('#AKEY1.Value *eq A1003')
Set Com(#ReturnCode) Value(ER)
Endif
Endcase
Endcase

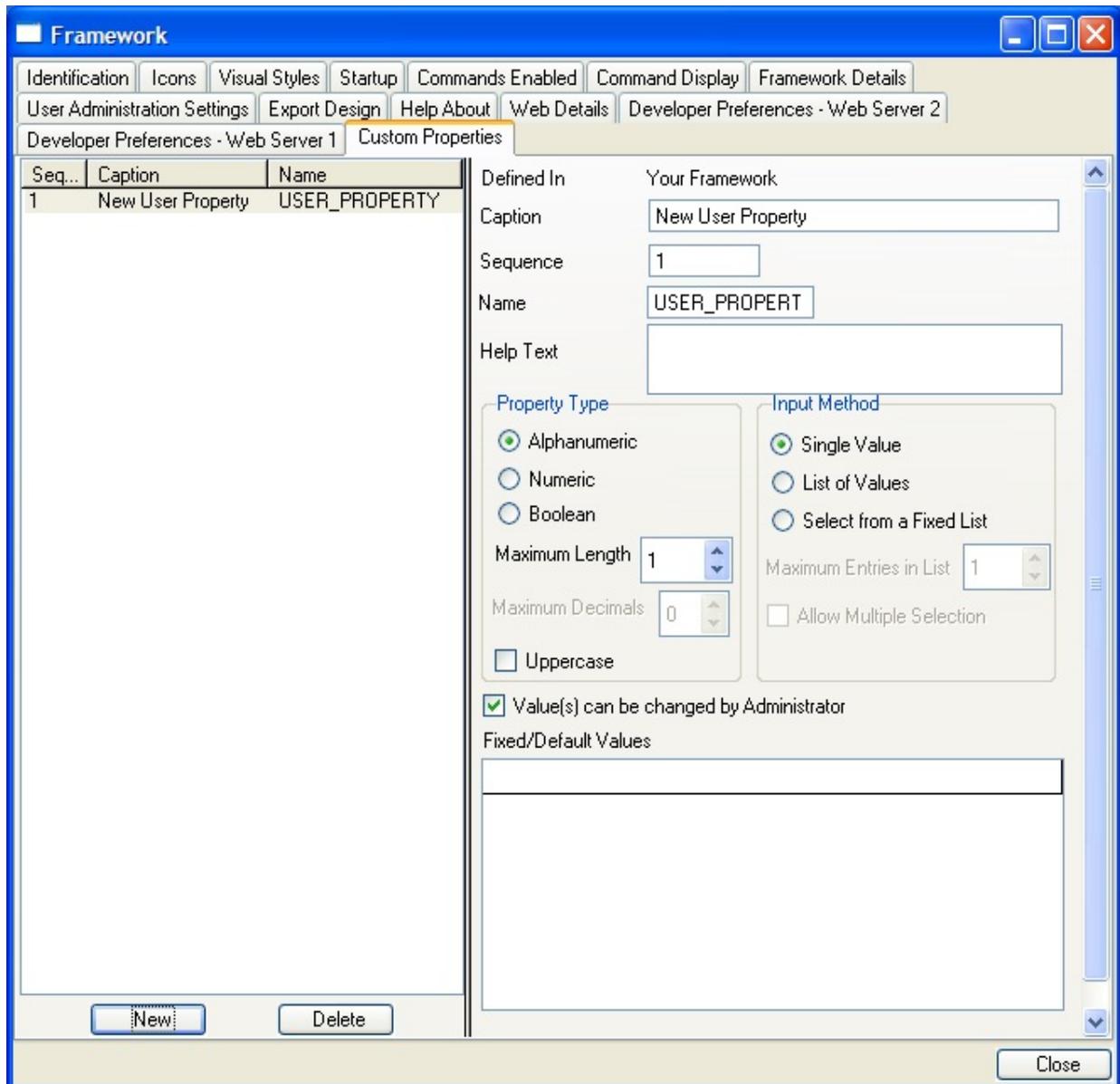
Endcase

Endroutine
```

Custom Properties

The Visual LANSA Framework uses a quite straightforward Framework, Application and Business Object model to define commercial computer systems.

You can extend this model by adding your own custom properties to the whole Framework, to Applications within it, or even to individual Business Objects:



The easiest way to understand the concept of custom properties is probably to consider this smattering of typical commercial application requirements:

- Only some users of this application can delete contracts.

- This user can display graphs upside down.
- Some users can only view contracts signed in certain states.
- All graphs presented should be rose colored.
- This user is restricted to working within this list of companies ...
- This user can view other employees at higher reward levels.
- All reports should always be printed locally.
- My preferred reporting currency is
- This system is installed in Japan?
- This user normally works in Korea.
- The URL of our company web site is www.mycompany.com (but it may change later).
- Does this system have a local Data Base?
- The preferred date format for this user is
- I need to associate an internal build number and date with my Framework to be shown in error situations.

There are many ways to address these types of common requirements. One way is to add them as custom properties to your Framework

For example, take the simple "The URL of our company web site is www.mycompany.com (but it may change later)" requirement. To make the company web site URL a soft coded value in your Framework you would do the following:

As a DESIGNER

You would define a Framework level custom property named COMPANYURL. It would most likely be of type Alphanumeric with a maximum length of 256. It would not be changeable by on site Administrators and have a default value of www.mycompany.com.

As a DEVELOPER

You would retrieve the value of custom property COMPANYURL into your programs to avoid hard coding of the company URL.

In both Windows and WAM applications you would retrieve the value like this:

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)
  Withname(COMPANYURL) AlphaValue(#COMP_URL)
```

As an ADMINISTRATOR

You would have nothing to do. You may notice that every time you create a user profile it has a property called "Company Web Site" associated with it that has value `www.mycompany.com` but you would not be able to change it. Only the Framework DESIGNER would be able to change the value.

As a more complex example consider the "Some users can only view contracts signed only in certain states" requirement. You could use custom properties to satisfy this requirement like this:

As a DESIGNER

You might define an application level custom property named `ALLOWSTATES`. It might be defined as a fixed alphanumeric list like this:

Value Caption to Display to User

ALL	Allow all States
CA	California Only
NY	New York Only
MN	Minnesota Only

As a DEVELOPER

You would retrieve the value of `ALLOWSTATES` into your programs and use it to control how your program behaves.

In both Windows and WAM applications you would retrieve the value like this:

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(A)  
Withname(ALLOWSTATE) AlphaValue(#ALL_STATE)
```

In either case `#ALL_STATE` would contain `ALL`, `CA`, `NY` or `MN` which your program would then use to control access to various contracts within your application.

As an ADMINISTRATOR

When you define a user you can switch to the Custom Properties tab of the current user and select which states the user can view contracts from. You would have to select from a list that look like this:

States allowed for reviewing contracts	Allow all States California Only New York Only Minnesota Only
--	--

These simple examples demonstrate the essence of custom properties. They of course have considerably more capability than this. For example you can specify the type as Alphanumeric, Numeric or Boolean and have multiple selection lists.

See also [Frequently Asked Questions about Custom Properties](#) and [Things to be careful with when using Custom Properties](#).

Defining Custom Properties

When defining a new property these options are displayed:

[Defined In](#)

[Caption](#)

[Sequence](#)

[Name](#)

[Help Text](#)

[Property Type](#)

[Maximum Decimals](#)

[Maximum Length](#)

[Uppercase](#)

[Input Method](#)

[Maximum Entries in List](#)

[Allow Multiple Selection](#)

[Value\(s\) can be changed by Administrator](#)

[Fixed / Default Values](#)

Frequently Asked Questions about Custom Properties

How do I handle lists of custom properties programmatically?

All custom properties have a special ".Count" property associated with them. This property tells you how many instances of a custom property currently exist.

For example, suppose you have specified a Framework level custom property named STATES where the Administrator could choose one or more states from a fixed list.

To determine how many states were chosen for the current user you do this:

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)
  Withname(STATES.Count) Numericvalue(#Tot_State)
```

#Tot_State now contains how many entries are currently in the list of states.

If you then wanted get the state values from the list you might code this:

```
Begin_Loop To(#Tot_sTate) Using(#Index)

  Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)
    Withname(STATES) Instance(#Index) AlphaValue(#State_Code)

  Add_Entry << possibly some user visible list >>

End_Loop
```

Why are the .COUNT values different by list type?

The .COUNT values returned to your program will vary by the type of custom property and the input method you have specified:

Type	Input Method	.COUNT Value / Comments
All	Single Value	Always 1
Alphanumeric, Numeric	List of Values	Always returned as the maximum number of entries allowed in the list. Your program needs to screen out which entries are "null" according to their values returned (e.g. blank or zero values may be considered

to be "null" entries within your application).

Boolean	List of Values	Not a valid custom property definition. You cannot define this type of custom property.
Alphanumeric, Numeric	Fixed List	The number of entries actually selected by the administrator (or by default). The default behavior is to select the first entry in the default fixed list specified by the designer.
Boolean	Fixed List	Always the maximum number of entries allowed in the fixed list. See next question as well.

What are Boolean Fixed Lists For?

Boolean fixed lists allow multiple Boolean values to be consolidated into a single custom property. This is often easier and more efficient than coding multiple single Boolean values and it keeps the options together.

For example, imagine you need to store these custom properties:

User is allowed Jump

User is allowed to Hop

User is allowed to Skip

User is allowed to Run

You can do this by defining four Boolean single value properties (e.g. named JUMP, HOP, SKIP and RUN). They would appear to the administrator like this:

User can Jump	<input type="checkbox"/> No
User can Hop	<input type="checkbox"/> No
User can Skip	<input type="checkbox"/> No
User can Run	<input type="checkbox"/> No

To retrieve these properties programmatically you would do something like this:

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
  Withname(JUMP) BooleanValue(#CAN_Jump)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
  Withname(HOP) BooleanValue(#CAN_Hop)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)
```

```
Withname(SKIP) BooleanValue(#CAN_Skip)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
Withname(RUN) BooleanValue(#CAN_Run)
```

You can also do this by using a Boolean fixed list containing 4 entries (e.g. named ACTIONS). This would appear to the administrator like this:

Actions User can Perform
<input type="checkbox"/> Jump
<input type="checkbox"/> Hop
<input type="checkbox"/> Skip
<input type="checkbox"/> Run

In this case each entry in the list represents a user state and would be programmatically accessed like this:

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
Withname(ACTIONS) Instance(1) BooleanValue(#CAN_Jump)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
Withname(ACTIONS) Instance(2) BooleanValue(#CAN_Hop)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
Withname(ACTIONS) Instance(3) BooleanValue(#CAN_Skip)
```

```
Invoke Method(#avFrameworkManager.avGetUserProperty) Atlevel(F)  
Withname(ACTIONS) Instance(4) BooleanValue(#CAN_Run)
```

What about Multilingual Systems?

Custom property captions, help text and fixed list user visible captions are all multilingual capable. You should be able to cause all aspects of your custom properties that are visible to administrators to be presented in the correct language by using the normal Framework translation procedures.

Why aren't property names unique in the Framework?

There are 2 main reasons for this:

- It means you can define a property like PRINTERNAME in both the Human Resources application and in the General Ledger application and have a different value specified for each application.
- You can create search lists for a property. For example you could look for a property named PRINTERNAME at the business object level. If it was not found then look at the application level. If it was still not found you might finally look at the Framework level.

My custom property does not seem to have the expected value?

Things that you should check first include:

- The custom property has default values.
- The Framework was saved and restarted since property was defined or changed.
- You are signed on as an end-user (i.e. Using UF_EXEC or equivalent).
- A level A (Application) property is being referenced by a command handler but there is no current application selected.
- A level B (Business Object) property is being referenced by a command handler but there is no current business object selected.

Things to be careful with when using Custom Properties

Test Custom Properties by Signing on as an End-user

You should always test your custom properties signed on as an end-user and using the UF_EXEC (or equivalent) entry point. Testing while signed on as designer (UF_DESGN or equivalent) or as an administrator (UF_ADMIN or equivalent) is not advisable.

Save and Restart the Framework after Altering Custom Properties

If you do wish to test your custom properties while using the Framework as a designer or administrator then you should always save and restart the Framework after making any changes to custom property definitions and before running applications that access them.

Always specify Default Value(s)

You should always specify a default value for any property. There are several reasons for doing this:

- If the property cannot be changed by the administrator then this is of course essential.
- If the administrator does not set or change the property for a specific user then the value is not stored with the user. The current default value is always used.
- If the property can be changed by the administrator then a good default will save the administrator time and reduce your support costs.

Spend time on Help Text

Spending three minutes on simple custom property help text now may save you a day of support time later.

Changing and Redeploying Custom Properties

If you change the definition of an existing custom property and/or remove values from a list associated with it and then (re)deploy your Framework definition you need to carefully consider the ramifications, especially if many users already have their own unique values and lists for the property saved.

Having too many Custom Properties

While there is no specific limit on how many custom properties you can define with in a Framework, having more than 100 would mean that your Framework might start to encounter some usability issues.

Custom User properties in WAMs

When using Custom User properties in WAMs, do not set or retrieve custom

user properties in the uHandleEvent routine. Allow the Framework to initialize the WAM first. Set or retrieve custom properties in the uExecute or other routines.

Writing queries over Visual LANSAs Framework objects

You can write query programs that report on the objects in the Visual LANSAs Framework (VLF).

You might use them to produce lists of objects in the framework, either to locate error situations or to assess development progress, in the same way as you might write queries over the LANSAs internal DC@ repository files. You can extract information from the framework and transform it into other forms or other media.

There are some important restrictions:

- Such programs should never be run in a production or end-user environments. They are tools for the developer in the developer's environment only.
- These programs can access internal VLF properties that are not usually accessible to command handlers or filters.
- The possibility exists that the VLF properties might change in the future. If this happens you might have to change your query programs, usually very slightly.
- These properties are not externally documented. If you don't understand a property or how it works then you may need to put a question onto a forum.
- The VLF properties are read only – they should never be written as you may corrupt your framework file.

[Getting Started](#)

[Using the First Query Example](#)

[Using the Second Query Example](#)

[Using the Third Query Example](#)

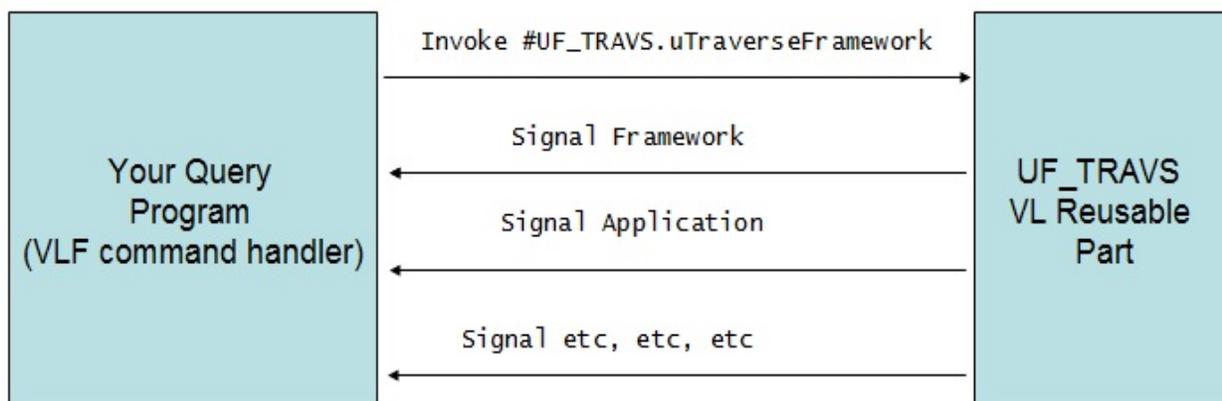
[Creating Your Own Queries](#)

[Other things worth knowing about Query Programs](#)

[Examples](#)

Getting Started

Create a VL reusable part named UF_TRVRS. Copy and paste the supplied example code from [UF_TRVRS - Signal VLF Objects](#) into it and compile it. UF_TRVRS is the engine that drives all queries. It traverses the current and signals back to your program when it finds different types of framework objects:



You start it by invoking `#UF_TRVRS.uTraverseFramework`. For each object encountered `uTraverseFramework` sends a signal back to your query program. The signals are not issued in any particular order. This process is very like parsing an XML document.

For example, if you wanted to make a list of all the captions of all the applications in a framework you would add a routine like this to your query program:

```
EvtRoutine Handling(#UF_TRVRS.Application) Reference(#TheApplication)
#Std_TextL := #TheApplication.UCaption
Add_Entry #MyListView
Endroutine
```

Using the First Query Example

Create a VL reusable part named UF_QRY01. Copy and paste the supplied example code from [UF_QRY01 - Simple Example of How to Listen For General Framework Objects And View Their Properties](#) into it.

All query examples are VLF command handlers and need to be snapped into the framework.

Create a VLF Application named “Programmer Tools” (say) and add a single business object named “Reports” (say) to it.

Delete all filters from the “Reports” business object.

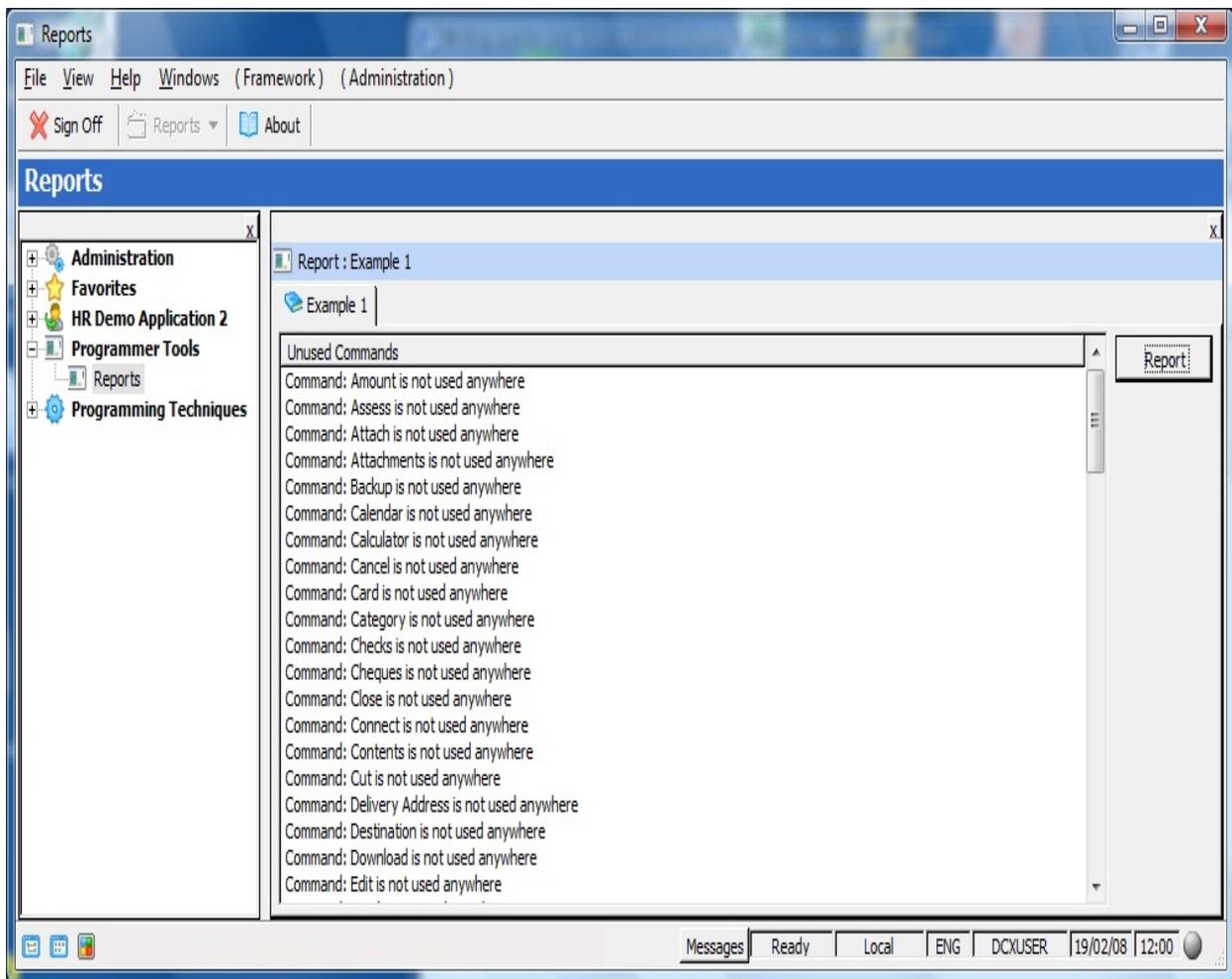
Change the “Command Tab Show All” option on the “Commands Display” tab to True.

Associate a command named “Example 1” with the business object as a business object level command (ie: not an instance level command). Make it the default command.

Snap reusable part UF_QRY01 in as the command handler.

Save and restart your framework and then execute the “Example 1” command.

Click the “Report” button. The resulting display should look something like this:



This report shows all the commands defined in your framework that are not used. This example report is not especially useful. It is designed to demonstrate how you can go about creating reports that match your exact needs.

Using the Second Query Example

Repeat the preceding process using [UF_QRY02 - Listening for General Framework Objects And View Their Properties](#) to create a reusable part named UF_QRY02 and snap in as the command handler for a tab/command handler named “Example 2”.

When you execute “Example 2” and click the report button the resulting display should look like this:

The screenshot shows a software application window titled "Reports". The window has a menu bar with "File", "View", "Help", "Windows (Framework)", and "(Administration)". Below the menu bar is a toolbar with "Sign Off", "Reports", and "About" buttons. The main area is titled "Reports" and contains a table with the following columns: "Parent Framework", "Parent Application", "Parent Business Object", and a description of the snap in component. The table lists various snap in components, including commands and business objects, with their corresponding framework, application, and business object identifiers. A sidebar on the left contains navigation options: "Administration", "Favorites", "HR Demo Application 2", "Organizations", "Resources", "Programmer Tools", "Reports", and "Programming Techniques". The status bar at the bottom shows "Messages", "Ready", "Local", "ENG", "DCXUSER", and the date/time "19/02/08 12:04".

Parent Framework	Parent Application	Parent Business Object	Description
			Framework DIME Shipped Framework VF_SHIPPED SH
SHIPPED_FRAMEWORK			Command About Framework Snap in windows handler
SHIPPED_FRAMEWORK			Command Exit Snap in windows handler VF_CH001 Sr
SHIPPED_FRAMEWORK			Command Sign Off Snap in windows handler VF_CH0C
SHIPPED_FRAMEWORK			Command Assistant Example 1 Snap in windows hand
SHIPPED_FRAMEWORK			Command Assistant Example 2 Snap in windows hand
SHIPPED_FRAMEWORK			Command Assistant Example 3 Snap in windows hand
SHIPPED_FRAMEWORK			Application Programming Techniques 78A26552E6E94
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Command About... Snap in windows handler VF_CH0I
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object The essential business object 9A475E
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	9A475843B9D5452D9A4926BDE8E	Command Details Snap in windows handler DF_T0023
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	9A475843B9D5452D9A4926BDE8E	Filter by SurnameSnap in windows filter DF_FILTER1 Sni
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object The CRUD business object 34C9245F
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	34C9245FB4C74621A893CCF65D	Command Details Snap in windows handler DF_DET21
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	34C9245FB4C74621A893CCF65D	Command New Snap in windows handler DF_DET21 S
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	34C9245FB4C74621A893CCF65D	Command Delete Snap in windows handler DF_DET21
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	34C9245FB4C74621A893CCF65D	Command Copy Snap in windows handler DF_DET21!
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	34C9245FB4C74621A893CCF65D	Filter CRUD blind filterSnap in windows filter DF_FILTER
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object Selected, Current or All Entries BD8D
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	BD8D1D64689F468E8FA84DA2DA	Command Selected Entry and Current Entry Snap in v
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	BD8D1D64689F468E8FA84DA2DA	Command All Entries Snap in windows handler DF_DE
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	BD8D1D64689F468E8FA84DA2DA	Filter Auto-fill with 10 employeesSnap in windows filte
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object Switching A1DC6AABA4534FFFB30B6914B5E
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	A1DC6AABA4534FFFB30B6914B5E	Command Switch to Employees Snap in windows hanc
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	A1DC6AABA4534FFFB30B6914B5E	Filter Sections by DepartmentSnap in windows filter C
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object The time fast part 96718E3E1A3D46
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	96718E3E1A3D462FA2B33C36EB3	Command Example 1 Snap in windows handler DF_TO
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	96718E3E1A3D462FA2B33C36EB3	Command Example 2 Snap in windows handler DF_TO
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C	96718E3E1A3D462FA2B33C36EB3	Command Example 3 Snap in windows handler DF_TO
SHIPPED_FRAMEWORK	78A26552E6E94627BBCC4DF3A248C		Business Object The data fast part DF_C57C740B9544

This report lists all the snap in components for every filter and command handler in the framework. This example report is not especially useful. It is designed to demonstrate how you can go about creating reports that match your exact needs.

Using the Third Query Example

Repeat the preceding process using [UF_QRY03 - Listening for Ramp Objects And View Their Properties](#) to create a reusable part named UF_QRY03 and snap in as the command handler for a tab/command handler named “Example 3”. When you execute “Example 3” and click the report button the resulting display should look like this:

Reports

File View Help Windows (Framework) (Administration)

Sign Off Reports About

Reports

- Administration
- Favorites
- HR Demo Application 2
- Programmer Tools
 - Reports
- Programming Techniques
 - Basic
 - Advanced
 - Fast Parts
 - Prompting

Report : Example 3

Example 1 Example 2 Example 3

	Member ID
RAMP Session Default Node Container 6B3680B83C834950AA496C7A9767AA4F 6B3680B83C834950AA496C7A9767AA4F	
RAMP Group Default Session 2 446F94117F754E76A6F7E6641FDE3170 446F94117F754E76A6F7E6641FDE3170	
Destination uWrkOutq Technical Caption RAMP Destination Screen (uWrkOutq-) Navigate Script ID CEC1AFC643B74762A9D872	64525AF82E9F43A9A90D02E0
Destination uWrkJobq Technical Caption RAMP Destination Screen (uWrkJobq-) Navigate Script ID A86CFA4C3F47430F8323D07	1830C2B16E4245888275083A
Destination uDisplay_EMPLOYEE_DETAILS Technical Caption RAMP Destination Screen (uDisplay_EMPLOYEE_DETAILS-) Naviga	510DBBACCDDFA4203A6CA02A
Junction uSignon Technical Caption RAMP Junction Screen (uSignonlongerstill-) Script ID D1446FB412D2419EB69CDA0A14A4DED3	D1446FB412D2419EB69CDA0A
Junction uOS400MainMenu Technical Caption RAMP Junction Screen (uOS400MainMenu-) Script ID 7BE254273F274EC4B53DAFB4	7BE254273F274EC4B53DAFB4
Junction uSignon Technical Caption RAMP Junction Screen (uSignon-) Script ID C93735F90186429D902FCE26F55085D0 Screen Nan	C93735F90186429D902FCE26
Junction uSignon Technical Caption RAMP Junction Screen (uSignon-) Script ID 74261C27BE6C42329C51192334A20905 Screen N	74261C27BE6C42329C511923
Junction uPSLSYSMenu Technical Caption RAMP Junction Screen (uPSLSYSMenu-) Script ID CA6B7455D96D4AFBB55BDA374A52F	CA6B7455D96D4AFBB55BDA3
Junction uLocateEMPLOYEE Technical Caption RAMP Junction Screen (uLocateEMPLOYEE-) Script ID 72120712124C4E95A19D42E4	72120712124C4E95A19D42E4
Special uLogin Message Screen Technical Caption RAMP Special Screen (uLogin Message Screen-)	41EBF70B74C84AFCB28FA67C
Script New Script Technical Caption Script (ELIMINATE_SCRIPT_6-uLogin Message Screen - Automatically eliminate this form)	EFFFD90AD3394662B25C9834
Script New Script Technical Caption Script (INVOKE_SCRIPT_6-uWrkOutq - Invoke this form from anywhere)	CEC1AFC643B74762A9D87201
Script New Script Technical Caption Script (RETURN_SCRIPT_6-uWrkOutq - Return to nearest junction form)	D482892FAC7240BF889D59E2
Script New Script Technical Caption Script (BUTTON_SCRIPT_6-uWrkOutq - Handle function keys and button usage)	0FA62F5B440C4537B12CE950
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_5-Navigate from signon ==> uOS400MainMenu)	5A849D4FC5BE4A6790F6B0E9
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_6-Navigate from uOS400MainMenu ==> signon)	2F299772DAC84412BA1441CB
Script New Script Technical Caption Script (INVOKE_SCRIPT_7-uWrkJobq - Invoke this form from anywhere)	A86CFA4C3F47430F8323D079
Script New Script Technical Caption Script (RETURN_SCRIPT_7-uWrkJobq - Return to nearest junction form)	52C4743E7FC3490DA8460C5E
Script New Script Technical Caption Script (BUTTON_SCRIPT_7-uWrkJobq - Handle function keys and button usage)	55318E1D60734531883A6E03
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_7-Navigate from signon ==> uOS400MainMenu)	EDADD8178CB3469094FCF3C
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_8-Navigate from uOS400MainMenu ==> signon)	8E7B95BE82964DF7B9FCFC9
Script New Script Technical Caption Script (INVOKE_SCRIPT_8-Display_EMPLOYEE_DETAILS - Invoke this form from anywhere)	40A6EA2C8ADC40D9A6AB31C
Script New Script Technical Caption Script (RETURN_SCRIPT_8-Display_EMPLOYEE_DETAILS - Return to nearest junction form)	C654195E47684984906471EB
Script New Script Technical Caption Script (BUTTON_SCRIPT_8-Display_EMPLOYEE_DETAILS - Handle function keys and button	E1248594DAC641C384E9BC4C
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_9-Navigate from uOS400MainMenu ==> uSignon)	E24DFDB445AE4600B1D106F1
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_10-Navigate from uSignon ==> uOS400MainMenu)	D4CE229B31504E7DA653489A
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_11-Navigate from uPSLSYSMenu ==> uOS400MainMenu)	05B7227C8F9B4B2AA4AD716C
Script New Script Technical Caption Script (NAVIGATE_SCRIPT_12-Navigate from uOS400MainMenu ==> uPSLSYSMenu)	0B272DF6074446088180F1EA

Report

Show Script

Show FKeys

```

/* Set up data fields on form uOS400MainMenu */
SETVALUE("Cmldine", "signoff");

/* Send the key required to navigate to signon */
SENDKEY(KeyEnter);

/* Check for arrival at signon */
if (! (CHECK_CURRENT_FORM("signon", "Unable to display form signon"))) return;

```

Messages Ready Local ENG DCXUSER 19/02/08 12:09

This report shows all the RAMP destination, special and junction scripts defined and allows you to display the script lines and the function keys associated with them. This example report is not especially useful. It is designed to demonstrate how you can go about creating reports that match your exact needs.

Creating Your Own Queries

Using the examples provided you should be able to start to create your own reports that match your exact needs. If you have a question or don't understand something use the VLF forum.

Other things worth knowing about Query Programs

Query program listens for the signals from UF_TRVRS for the types of object it is interested in, and stores information about the objects.

The query program can store the object information any way it chooses: It can store object references in a collection, or in a List or grid (using the .RelatedReference property of the current item). Or it can get the object property information at the time it receives the signal, and store it in fields in an ordinary list.

There are two kinds of framework object:

- Normal objects with names like VF_FPnnn (e.g. VF_FP003). There are multiple instances of these objects, and if you have the object reference, it is a reference to a particular instance. You automatically have access to all the properties of the instance.
- Member objects with names like VF_FPMnn (e.g. VF_FPM09). These objects were designed this way for performance reasons. In this case there is only one actual instance, and the properties of all instances are stored internally inside the single object instance. To get the properties of a particular instance, you must first set a property called CurrentMemberGUID to the value for that instance.

The main framework objects in these examples are:

VF_FP001	The framework
VF_FP002	Application
VF_FP003	Business Object
VF_FPM08	Application/Business Object link
VF_FPM09	Command Definition
VF_FPM10	Command Reference (link between a command and a business object or application or framework) This contains command handler information.
VF_FPM14	Filter

The main RAMP objects are:

VF_FP025	RAMP Container
VF_FP026	Session
VF_FPM27	Destination
VF_FPM28	Junction
VF_FPM29	Special
VF_FPM30	Script

By exploring these objects using the Visual LANS A F2=Feature Help option you can discover the properties that they contain. If you have questions about the properties please post a message to the VLF forum.

Examples

UF_TRVRS - Signal VLF Objects

UF_QRY01 - Simple Example of How to Listen For General Framework Objects And View Their Properties

UF_QRY02 - Listening for General Framework Objects And View Their Properties

UF_QRY03 - Listening for Ramp Objects And View Their Properties

UF_TRVRS - Signal VLF Objects

```
=====
*
* Component   : UF_TRVRS
* Type       : Reusable Component
* Ancestor   : PRIM_OBJT
*
* PLEASE NOTE: This UF_ (User Framework) component is the shipped
version. You
* may choose to modify it. You should do this by copying the source
* code of this component into your own component and then change
* the copied version. This will prevent the accidental loss of your
* changes if you upgrade your Visual LANSA framework version. Refer
* to the end of this component for more details about making your
* own version of this component.
*
* This is example code only - No warranty is expressed or implied.
* Neither this program, nor any derivative of it, should be ever be used in
* production or end-user environments.
```

```
=====
*
* This reusable part can be used to signal VLF objects - See UF_QRY01, 02
03
```

```
=====
FUNCTION OPTIONS(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_OBJT)
```

```
Define_Com Class(#VF_SY001) Name(#USYSTEM) Reference(*dynamic)
scope(*Application)
```

```
Define_Com Class(#VF_FP001) Name(#UFRAMEWORK)
Reference(*dynamic) scope(*Application)
```

```
Define_Com #VF_AC009 #TempVF_AC009 Reference(*Dynamic)
```

```
Define_Com #VF_FP001 #TempVF_FP001 Reference(*Dynamic)
Define_Com #VF_FP002 #TempVF_FP002 Reference(*Dynamic)
Define_Com #VF_FP003 #TempVF_FP003 Reference(*Dynamic)
```

```
Define_Com #VF_FPM09 #TempVF_FPM09 Reference(*Dynamic)
Define_Com #VF_FPM10 #TempVF_FPM10 Reference(*Dynamic)
Define_Com #VF_FPM14 #TempVF_FPM14 Reference(*Dynamic)
```

```
Define_Com #VF_FP025 #TempVF_FP025 Reference(*Dynamic)
Define_Com #VF_FP026 #TempVF_FP026 Reference(*Dynamic)
```

```
Define_Com #VF_FPM27 #TempVF_FPM27 Reference(*Dynamic)
Define_Com #VF_FPM28 #TempVF_FPM28 Reference(*Dynamic)
Define_Com #VF_FPM29 #TempVF_FPM29 Reference(*Dynamic)
Define_Com #VF_FPM30 #TempVF_FPM30 Reference(*Dynamic)
```

*

=====

* Events signalled back to user application for each type of object found

*

=====

```
Define_evt Framework
define_map *input #VF_FP001 #Reference Pass(*By_Reference)
```

```
Define_evt Application
define_map *input #VF_FP002 #Reference Pass(*By_Reference)
define_map *input #VF_FP001 #VisParent1 Pass(*By_Reference)
```

```
Define_evt BusinessObject
define_map *input #VF_FP003 #Reference Pass(*By_Reference)
define_map *input #VF_FP001 #VisParent1 Pass(*By_Reference)
define_map *input #VF_FP002 #VisParent2 Pass(*By_Reference)
```

```
Define_evt CommandDefinition
define_map *input #VF_FPM09 #Reference Pass(*By_Reference)
```

```
Define_evt CommandReference
```

```
define_map *input #VF_FPM10 #Reference Pass(*By_Reference)
define_map *input #VF_FPM09 #CommandDefinition Pass(*By_Reference)
define_map *input #VF_FP001 #VisParent1 Pass(*By_Reference)
define_map *input #VF_FP002 #VisParent2 Pass(*By_Reference)
define_map *input #VF_FP003 #VisParent3 Pass(*By_Reference)
```

Define_evt Filter

```
define_map *input #VF_FPM14 #Reference Pass(*By_Reference)
define_map *input #VF_FP001 #VisParent1 Pass(*By_Reference)
define_map *input #VF_FP002 #VisParent2 Pass(*By_Reference)
define_map *input #VF_FP003 #VisParent3 Pass(*By_Reference)
```

define_evt RAMPContainer

```
define_map *input #VF_FP025 #Reference Pass(*By_Reference)
```

define_evt RAMPSession

```
define_map *input #VF_FP026 #Reference Pass(*By_Reference)
```

define_evt RAMPDestination

```
define_map *input #VF_FPM27 #Reference Pass(*By_Reference)
```

define_evt RAMPJunction

```
define_map *input #VF_FPM28 #Reference Pass(*By_Reference)
```

define_evt RAMPSpecial

```
define_map *input #VF_FPM29 #Reference Pass(*By_Reference)
```

define_evt RAMPScript

```
define_map *input #VF_FPM30 #Reference Pass(*By_Reference)
```

*

=====

* This is the method exposed to the user application

*

=====

Mthroutine Name(uTraverseFramework)

```
Invoke #Com_Owner.uEnumerateOBJECT Reference(#uFramework)
Parent(*null)
```

```
Invoke #Com_Owner.uEnumerateOBJECT
Reference(#uSystem.uNodeContainer) Parent(*null)
```

```
Endroutine
```

```
*
```

```
=====
```

```
* This method is internal and not exposed
```

```
*
```

```
=====
```

```
Mthroutine Name(uEnumerateOBJECT) Access(*PRIVATE)
Define_Map For(*input) Class(#VF_AC001) Name(#Reference)
Pass(*By_Reference)
Define_Map For(*input) Class(#VF_AC001) Name(#Parent)
Pass(*By_Reference)
```

```
Define_com #vf_elindx #EnumIndex
Define_com #vf_elmbri #EnumMemberGUID
```

```
If_ref #Reference is_not(*null)
```

```
* Handle this object bvy signalling each type found
```

```
Case #Reference.uClass
```

```
* Framework
```

```
When (= VF_FP001)
set_ref #TempVF_FP001 (*Dynamic #Reference)
Signal Framework Reference(#TempVF_FP001)
```

```
* Application
```

```
When (= VF_FP002)
set_ref #TempVF_FP002 (*Dynamic #Reference)
```

```
set_ref #TempVF_FP001 (*Dynamic #TempVF_FP002.uAuthorityParent)
Signal Application Reference(#TempVF_FP002)
VisParent1(#TempVF_FP001)
```

* Business Object

```
When (= VF_FP003)
set_ref #TempVF_FP003 (*Dynamic #Reference)
set_ref #TempVF_FP002 (*Dynamic #TempVF_FP003.uAuthorityParent)
set_ref #TempVF_FP001 (*Dynamic #TempVF_FP002.uAuthorityParent)
```

```
Signal BusinessObject Reference(#TempVF_FP003)
VisParent1(#TempVF_FP001) VisParent2(#TempVF_FP002)
```

* RAMP Container

```
When (= VF_FP025)
set_ref #TempVF_FP025 (*Dynamic #Reference)
Signal RAMPContainer Reference(#TempVF_FP025)
```

* RAMP Session

```
When (= VF_FP026)
set_ref #TempVF_FP026 (*Dynamic #Reference)
Signal RAMPSession Reference(#TempVF_FP026)
```

Endcase

* Enumerate all members

```
If_ref #Reference.ecMemberManagers is_not(*null)
For #Manager #Reference.ecMemberManagers
```

```
#EnumIndex := 0
```

```
Dowhile (#Manager.Enumerate_MEMBERS(#Reference #EnumIndex
#EnumMemberGUID 7))
```

```
#Manager.Currentmemberguid := #EnumMemberGUID
```

Case #Manager.VF_FPNNNClass

* Command Definition

When (= VF_FP009)

set_ref #TempVF_FPM09 (*Dynamic #Manager)

Signal CommandDefinition Reference(#TempVF_FPM09)

* Command reference

When (= VF_FP010)

set_ref #TempVF_FPM10 (*Dynamic #Manager)

#uFrameWork.VF_FP009Manager.CurrentmemberGUID :=

#TempVF_FPM10.uCommandGUID

set_ref #TempVF_AC009 (*Dynamic #TempVF_FPM10.uLinkedOwner)

If_Ref Com(#TempVF_AC009) Is(*INSTANCE_OF #VF_FP001)

* Framework Command

set_ref #TempVF_FP003 *null

set_ref #TempVF_FP002 *null

set_ref #TempVF_FP001 (*Dynamic #TempVF_AC009)

ENDIF

If_Ref Com(#TempVF_AC009) Is(*INSTANCE_OF #VF_FP002)

* Application Command

set_ref #TempVF_FP003 *null

set_ref #TempVF_FP002 (*Dynamic #TempVF_AC009)

set_ref #TempVF_FP001 (*Dynamic #TempVF_FP002.uAuthorityParent)

ENDIF

If_Ref Com(#TempVF_AC009) Is(*INSTANCE_OF #VF_FP003)

* Business Object Command

set_ref #TempVF_FP003 (*Dynamic #TempVF_AC009)

set_ref #TempVF_FP002 (*Dynamic #TempVF_FP003.uAuthorityParent)

set_ref #TempVF_FP001 (*Dynamic #TempVF_FP002.uAuthorityParent)

ENDIF

Signal CommandReference Reference(#TempVF_FPM10)
CommandDefinition(#uFrameWork.VF_FP009Manager)
VisParent1(#TempVF_FP001) VisParent2(#TempVF_FP002)
VisParent3(#TempVF_FP003)

* Filter

When (= VF_FP014)
set_ref #TempVF_FPM14 (*Dynamic #Manager)
set_ref #TempVF_FP003 (*Dynamic #Reference)
set_ref #TempVF_FP002 (*Dynamic #TempVF_FP003.uAuthorityParent)
set_ref #TempVF_FP001 (*Dynamic #TempVF_FP002.uAuthorityParent)

Signal Filter Reference(#TempVF_FPM14) VisParent1(#TempVF_FP001)
VisParent2(#TempVF_FP002) VisParent3(#TempVF_FP003)

* RAMP Destination

When (= VF_FP027)
set_ref #TempVF_FPM27 (*Dynamic #Manager)
Signal RAMPDestination Reference(#TempVF_FPM27)

* RAMP Junction

When (= VF_FP028)
set_ref #TempVF_FPM28 (*Dynamic #Manager)
Signal RAMPJunction Reference(#TempVF_FPM28)

* RAMP Special

When (= VF_FP029)
set_ref #TempVF_FPM29 (*Dynamic #Manager)
Signal RAMPSpecial Reference(#TempVF_FPM29)

* RAMP Script

When (= VF_FP030)
set_ref #TempVF_FPM30 (*Dynamic #Manager)

Signal RAMPScript Reference(#TempVF_FPM30)

Endcase

Endwhile

Endfor

Endif

* Enumerate all children recursively

If_ref #Reference.ecchildcollection is_not(*null)

For #Child in(#Reference.ecchildcollection)

Invoke #Com_Owner.uEnumerateOBJECT Reference(#Child)

Parent(#Reference)

Endfor

Endif

Endif

Endroutine

END_COM

UF_QRY01 - Simple Example of How to Listen For General Framework Objects And View Their Properties

```
=====
*
* Component   : UF_QRY01
* Type       : Reusable Component
* Ancestor   : VF_AC010 (Command Handler)
*
* PLEASE NOTE: This UF_ (User Framework) component is the shipped
version. You
* may choose to modify it. You should do this by copying the source
* code of this component into your own component and then change
* the copied version. This will prevent the accidental loss of your
* changes if you upgrade your Visual LANSA framework version. Refer
* to the end of this component for more details about making your
* own version of this component.
*
* This is example code only - No warranty is expressed or implied.
* Neither this program, nor any derivative of it, should be ever be used in
* production or end-user environments.
```

```
=====
*
* This is the simplest example of how to listen for general Framework objects
and view their properties
```

```
=====
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #VF_AC010) Height(336)
Layoutmanager(#ATLM_2) Width(552)
```

```
=====
* Simple Field and Group Definitions
```

```
=====
*
```

=====
* Component definitions

*

=====
* Body and Button arrangement panels

Define_Com Class(#PRIM_PANL) Name(#BUTTON_PANEL)
Displayposition(2) Height(336) Layoutmanager(#BUTTON_FLOW)
Left(464) Parent(#COM_OWNER) Tabposition(2) Tabstop(False) Top(0)
Width(88)

Define_Com Class(#PRIM_PANL) Name(#BODY_HEAD)
Displayposition(1) Height(336) Layoutmanager(#ATLM_1) Left(0)
Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(0)
Verticalscroll(True) Width(464)

* Attachment and flow layout managers

Define_Com Class(#PRIM_ATLM) Name(#MAIN_LAYOUT)
Define_Com Class(#PRIM_ATLI) Name(#BUTTON_ATTACH)
Attachment(Right) Manage(#BUTTON_PANEL) Parent(#MAIN_LAYOUT)
Define_Com Class(#PRIM_FWLM) Name(#BUTTON_FLOW)
Direction(TopToBottom) Flowoperation(Center) Marginbottom(4)
Marginleft(4) Marginright(4) Margintop(4) Spacing(4) Spacingitems(4)

Define_Com Class(#PRIM_ATLI) Name(#BODY_ATTACH)
Attachment(Center) Manage(#BODY_HEAD) Parent(#MAIN_LAYOUT)

* The report button

Define_Com Class(#PRIM_PHBN) Name(#PHBN_REPORT)
Buttondefault(True) Caption('Report') Displayposition(1) Left(4)
Parent(#BUTTON_PANEL) Tabposition(1) Top(4)
Define_Com Class(#PRIM_FWLI) Name(#FWLI_SAVE_BUTTON)
Manage(#PHBN_REPORT) Parent(#BUTTON_FLOW)

* The Traverse component

Define_Com Class(#UF_TRVRS) Name(#UF_TRVRS)

```
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1) Marginbottom(2)
Marginleft(2) Marginright(2) Margintop(2)
```

```
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center)
Parent(#ATLM_1)
```

```
Define_Com Class(#PRIM_ATLM) Name(#ATLM_2)
```

```
Define_Com Class(#PRIM_ATLI) Name(#ATLI_2) Attachment(Center)
Manage(#BODY_HEAD) Parent(#ATLM_2)
```

```
Define_Com Class(#PRIM_ATLI) Name(#ATLI_3) Attachment(Right)
Manage(#BUTTON_PANEL) Parent(#ATLM_2)
```

```
Define_Com Class(#Prim_kCol<#VF_FP003 #VF_ELIDN>)
Name(#collBusObj)
```

* List of the GUIDs of all the Commands

```
Def_List Name(#ListCmdGd) Fields(#cmdGUID #cmdUID) Type(*Working)
Entrys(*MAX)
```

```
define #cmdGUID reffld(#vf_elidn) desc('GUID for a command definition')
```

```
define #cmdUID reffld(#vf_elidn) desc('uIdentifier for a command definition')
```

* List of the GUIDs of all the Command References

```
Def_List Name(#ListCmRGd) Fields(#cmdRefGUD #cmdRefUID)
Type(*Working) Entrys(*MAX)
```

```
define #cmdRefGUD reffld(#vf_elidn) desc('GUID for a command reference')
```

```
define #cmdRefUID reffld(#vf_elidn) desc('uIdentifier for a command
definition')
```

```
Define_com #VF_FPM09 #TheVF_FPM09 Reference(*Dynamic)
```

```
Define_com #VF_FPM10 #TheVF_FPM10 Reference(*Dynamic)
```

```
Define_Com Class(#PRIM_LTVW) Name(#LTVW_1) Componentversion(2)
Displayposition(1) Fullrowselect(True) Height(332) Left(2)
```

```
Parent(#BODY_HEAD) Showsortarrow(True) Tabposition(1) Top(2)
Width(460)
```

```
Define_Com Class(#PRIM_ATLI) Name(#ATLI_4) Attachment(Center)
Manage(#LTVW_1) Parent(#ATLM_1)
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_1) Caption('Unused
Commands') Captiontype(Caption) Displayposition(1) Parent(#LTVW_1)
Source(#DF_ELTXTL) Width(100)
```

*

=====

* Events Definitions

*

=====

*

=====

* Property Definitions

*

=====

*

=====

* Method Definitions

*

=====

* -----

* Handle Command Execution

* -----

Mthroutine Name(uExecute) Options(*REDEFINE)

* The return code field and testing condition

Define #Ret_Code reffld(#IO\$STS)

Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')

* Do any execution logic defined in the ancestor

Invoke #Com_Ancesor.uExecute

Endroutine

*

=====

* Subroutines

*

=====
*

=====
* Event Handlers
*

=====
* -----
* Handle the save button
* -----

EVTROUTINE HANDLING(#PHBN_REPORT.Click)

* Clear Lists

clr_list #listCmdGD
clr_list #listCmRGD

* Tell the Traverse reusable part to read through the entire system in no particular order

* This program listens for the signals for each new object,

* Collect all the information

invoke #UF_TRVRS.uTraverseFramework

* Now display a list of any commands that are not referenced by anything

invoke #Com_Owner.Check_Unused

ENDROUTINE

* Listen for a Business Object - An example of storing ordinary objects

EVTROUTINE HANDLING(#UF_TRVRS.BusinessObject)

Reference(#TempVF_FP003)

* Store all the business objects in a collection

Set_ref Com(#collBusObj<#TempVF_FP003.uIdentifier>)
to(#TempVF_FP003)

endroutine

* Listen for a Command Definition Object

```
EVTROUTINE HANDLING(#UF_TRVRS.CommandDefinition)
Reference(#TempVF_FPM09)
```

* All the Command Definitions are stored as internal members of a single object, #VF_FPM09.

* There is only one object reference that needs to be stored
set_ref #TheVF_FPM09 #TempVF_FPM09

* But to access the information about a particular Command Definition, we need to know

* the value of a special property called .CurrentMemberGUID.

* So, store that value in a list

```
Change #cmdGUID #TempVF_FPM09.CurrentMemberGUID
Change #cmdUID #TempVF_FPM09.uIdentifier
```

```
Add_entry #ListCmdGD
```

endroutine

* Listen for a Command Usage (Command Reference) Object

```
EVTROUTINE HANDLING(#UF_TRVRS.CommandReference)
Reference(#TempVF_FPM10) CommandDefinition(#TempVF_FPM09)
```

* All the Command References are stored as internal members of a single object, #VF_FPM10.

* There is only one object that needs to be stored
set_ref #TheVF_FPM10 #TempVF_FPM10

* But to access the information about a particular Command Reference, we need to know

* the value of a special property called .CurrentMemberGUID.

* So, store that value in a list

```
Change #cmdRefGUID #TempVF_FPM10.CurrentMemberGUID
```

* Store the uIdentifier of the command that is referred to

```
Change #cmdRefUID #TempVF_FPM09.uIdentifier
```

```
Add_entry #ListCmRGd
```

```
ENDROUTINE
```

* Check which commands are not used anywhere

```
mthroutine Check_Unused
```

```
clr_list #LTVW_1
```

```
selectlist #ListCmdGD
```

* Which commands do not have a Command Reference that uses them?

```
Loc_Entry In_List(#ListCmRGd) Where(#cmdUID *eq #cmdrefUID)
```

```
if_status is_not(*Okay)
```

* Get Details of the command definition being processed

* (Set the GUID property first)

```
set #TheVF_FPM09 CurrentMemberGUID(#cmdGUID)
```

```
#df_eltxtl := 'Command: ' + #TheVF_FPM09.uCaption + ' is not used  
anywhere'
```

* To see what other command definition properties you can view, click on

#TheVF_FPM09 above and press F2, and look at the properties of the VF_FPM09 class

```
add_entry #LTVW_1
endif
endselect
endroutine
End_Com
```

UF_QRY02 - Listening for General Framework Objects And View Their Properties

```
=====
*
* Component   : UF_QRY02
* Type       : Reusable Component
* Ancestor   : VF_AC010 (Command Handler)
*
* PLEASE NOTE: This UF_ (User Framework) component is the shipped
version. You
* may choose to modify it. You should do this by copying the source
* code of this component into your own component and then change
* the copied version. This will prevent the accidental loss of your
* changes if you upgrade your Visual LANSA framework version. Refer
* to the end of this component for more details about making your
* own version of this component.
*
* This is example code only - No warranty is expressed or implied.
* Neither this program, nor any derivative of it, should be ever be used in
* production or end-user environments.
```

```
=====
*
* This demonstrates how to listen for general Framework objects and view
their properties
```

```
=====
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #VF_AC010) Height(336)
Layoutmanager(#ATLM_2) Width(552)
```

```
=====
* Simple Field and Group Definitions
```

```
=====
*
```

=====
* Component definitions

*

=====
* Body and Button arrangement panels

Define_Com Class(#PRIM_PANL) Name(#BUTTON_PANEL)
Displayposition(2) Height(336) Layoutmanager(#BUTTON_FLOW)
Left(464) Parent(#COM_OWNER) Tabposition(2) Tabstop(False) Top(0)
Width(88)

Define_Com Class(#PRIM_PANL) Name(#BODY_HEAD)
Displayposition(1) Height(336) Layoutmanager(#ATLM_1) Left(0)
Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(0)
Verticalscroll(True) Width(464)

* Attachment and flow layout managers

Define_Com Class(#PRIM_ATLM) Name(#MAIN_LAYOUT)
Define_Com Class(#PRIM_ATLI) Name(#BUTTON_ATTACH)
Attachment(Right) Manage(#BUTTON_PANEL) Parent(#MAIN_LAYOUT)
Define_Com Class(#PRIM_FWLM) Name(#BUTTON_FLOW)
Direction(TopToBottom) Flowoperation(Center) Marginbottom(4)
Marginleft(4) Marginright(4) Margintop(4) Spacing(4) Spacingitems(4)

Define_Com Class(#PRIM_ATLI) Name(#BODY_ATTACH)
Attachment(Center) Manage(#BODY_HEAD) Parent(#MAIN_LAYOUT)

* The report button

Define_Com Class(#PRIM_PHBN) Name(#PHBN_REPORT)
Buttondefault(True) Caption('Report') Displayposition(1) Left(4)
Parent(#BUTTON_PANEL) Tabposition(1) Top(4)
Define_Com Class(#PRIM_FWLI) Name(#FWLI_SAVE_BUTTON)
Manage(#PHBN_REPORT) Parent(#BUTTON_FLOW)

* The Traverse component

Define_Com Class(#UF_TRVRS) Name(#UF_TRVRS)

Define_Com Class(#PRIM_GRID) Name(#LTVW_1) Displayposition(1)
Height(332) Left(2) Parent(#BODY_HEAD) Rowresize(True)
Showsortarrow(True) Tabposition(1) Top(2) Width(460)
Define_Com Class(#PRIM_GDCL) Name(#LVCL_1) Caption('Parent
Framework') Captiontype(Caption) Displayposition(1) Parent(#LTVW_1)
Source(#FP_EKEY1) Width(12)
Define_Com Class(#PRIM_GDCL) Name(#LVCL_2) Caption('Parent
Application') Captiontype(Caption) Displayposition(2) Parent(#LTVW_1)
Source(#FP_EKEY2) Width(12)
Define_Com Class(#PRIM_GDCL) Name(#LVCL_3) Caption('Parent
Business Object') Captiontype(Caption) Displayposition(3) Parent(#LTVW_1)
Source(#FP_EKEY3) Width(17)
Define_Com Class(#PRIM_GDCL) Name(#LVCL_4) Displayposition(4)
Parent(#LTVW_1) Source(#DF_ELTXTL) Widthtype(Remainder)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1) Marginbottom(2)
Marginleft(2) Marginright(2) Margintop(2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center)
Parent(#ATLM_1)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_2) Attachment(Center)
Manage(#BODY_HEAD) Parent(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_3) Attachment(Right)
Manage(#BUTTON_PANEL) Parent(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_4) Attachment(Center)
Manage(#LTVW_1) Parent(#ATLM_1)

* Framework - class #VF_FP001 #TempVF_FP001 Reference(*Dynamic)
* Application - class #VF_FP002 #TempVF_FP002 Reference(*Dynamic)
* Business Object - class #VF_FP003 #TempVF_FP003 Reference(*Dynamic)
* Command - class #VF_FPM09 #TempVF_FPM09 Reference(*Dynamic)
* Command Usage - class #VF_FPM10 #TempVF_FPM10
Reference(*Dynamic)
* Filter - class #VF_FPM14 #TempVF_FPM14 Reference(*Dynamic)

*

=====

* Events Definitions

*

=====

*

=====

* Property Definitions

*

=====

*

=====

* Method Definitions

*

=====

* -----

* Handle Command Execution

* -----

Mthroutine Name(uExecute) Options(*REDEFINE)

* The return code field and testing condition

Define #Ret_Code reffld(#IO\$STS)

Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')

* Do any execution logic defined in the ancestor

Invoke #Com_Ancesor.uExecute

Endroutine

*

=====

* Subroutines

*

=====

*

=====

* Event Handlers

*

=====

* -----

* Handle the save button

* -----

EVTROUTINE HANDLING(#PHBN_REPORT.Click)

* Tell the Traverse reusable part to read through the entire system in no particular order

* This program listens for the signals for each new object,

clr_list #LTVW_1

invoke #UF_TRVRS.uTraverseFramework

ENDROUTINE

* Listen for a Framework Object

EVTROUTINE HANDLING(#UF_TRVRS.Framework)

Reference(#TempVF_FP001)

* Output generic details

Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('Framework')
uObject(#TempVF_FP001)

add_entry #LTVW_1

* Store a reference to the object against the list line

Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FP001)

ENDROUTINE

* Listen for an Application Object

EVTROUTINE HANDLING(#UF_TRVRS.Application)
Reference(#TempVF_FP002) VisParent1(#TempVF_FP001)

* Output generic details

Invoke #Com_Owner.uOutputGeneric uLevel(2) uType('Application')
uObject(#TempVF_FP002) VisParent1(#TempVF_FP001)

* For more details of the application object, click on #TempVF_FP002 above
and press F2, and look at the properties of class VF_FP002

add_entry #LTVW_1

* Store a reference to the object against the list line

Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FP002)

ENDROUTINE

* Listen for a Business Object Object

EVTROUTINE HANDLING(#UF_TRVRS.BusinessObject)
Reference(#TempVF_FP003) VisParent1(#TempVF_FP001)
VisParent2(#TempVF_FP002)

* Output generic details

Invoke #Com_Owner.uOutputGeneric uLevel(3) uType('Business Object')
uObject(#TempVF_FP003) VisParent1(#TempVF_FP001)
VisParent2(#TempVF_FP002)

* For more details of the business object object, click on #TempVF_FP003
above and press F2, and look at the properties of class VF_FP003

add_entry #LTVW_1

* Store a reference to the object against the list line

Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FP003)

ENDROUTINE

* Listen for a Filter Object

```
EVTROUTINE HANDLING(#UF_TRVRS.Filter)
Reference(#TempVF_FPM14) VisParent1(#TempVF_FP001)
VisParent2(#TempVF_FP002) VisParent3(#TempVF_FP003)
```

* Output generic details

```
Invoke #Com_Owner.uOutputGeneric uLevel(3) uType('Business Object')
uObject(#TempVF_FP003) VisParent1(#TempVF_FP001)
VisParent2(#TempVF_FP002) VisParent3(#TempVF_FP003)
```

* Specific details

* For more details of the filter object, click on #TempVF_FPM14 above and press F2, and look at the properties of class VF_FPM14

```
#DF_ELTXTL := 'Filter ' + #TempVF_FPM14.uCaption + 'Snap in windows
filter ' + #TempVF_FPM14.uFilterName + ' Snap In Web function ' +
#TempVF_FPM14.uWebFilterFunction + ' ' + 'Snap in wam filter ' +
#TempVF_FPM14.uWAMComponent
```

```
add_entry #LTVW_1
```

* Store a reference to the object against the list line

```
Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FPM14)
```

```
ENDROUTINE
```

* Listen for a Command Usage (Command Reference) Object

```
EVTROUTINE HANDLING(#UF_TRVRS.CommandReference)
Reference(#TempVF_FPM10) CommandDefinition(#TempVF_FPM09)
VisParent1(#TempVF_FP001) VisParent2(#TempVF_FP002)
VisParent3(#TempVF_FP003)
```

* Output generic details

```
Invoke #Com_Owner.uOutputGeneric uLevel(3) uType('Command
```

Reference') VisParent1(#TempVF_FP001) VisParent2(#TempVF_FP002)
VisParent3(#TempVF_FP003)

* Specific details

* Some of the details are on the Command Object (VF_FPM09)

* Most of the details are on the Command Reference Object (VF_FPM10)

* For more details of the Command and command reference objects, click on #TempVF_FPM09 or #TempVF_FPM10 above and press F2, and look at the properties of classes VF_FPM09 and VF_FPM10

```
#DF_ELTXTL := 'Command ' + #TempVF_FPM09.uCaption + ' Snap in  
windows handler ' + #TempVF_FPM10.uHandlerName + ' Snap In Web  
function ' + #TempVF_FPM10.uWebHandlerFunction + ' ' + ' Snap in wam  
handler ' + #TempVF_FPM10.uWAMComponent + ' uWAM' +  
#TempVF_FPM10.uWAM
```

```
add_entry #LTVW_1
```

* Store a reference to the object against the list line

```
Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FPM10)
```

ENDROUTINE

* Output a line of description of the object

```
mthroutine uOutputGeneric  
define_map *input #std_num #uLevel  
define_map *input #df_elmsg #uType  
Define_Map For(*input) Class(#vf_fp001) Name(#VisParent1)  
Mandatory(*NULL) Pass(*BY_REFERENCE)  
Define_Map For(*input) Class(#vf_fp002) Name(#VisParent2)  
Mandatory(*NULL) Pass(*BY_REFERENCE)  
Define_Map For(*input) Class(#vf_fp003) Name(#VisParent3)  
Mandatory(*NULL) Pass(*BY_REFERENCE)  
Define_Map For(*input) Class(#vf_ac001) Name(#uObject)  
Mandatory(*NULL) Pass(*BY_REFERENCE)
```

* Parent Details

Change (#FP_EKEY1 #FP_EKEY2 #FP_EKEY3) *null

* Parent 1 details

if_ref #VisParent1 is_not(*null)

Change #FP_EKEY1 #VisParent1.uUserObjectType

endif

* Parent 2 details

if_ref #VisParent2 is_not(*null)

Change #FP_EKEY2 #VisParent2.uUserObjectType

endif

* Parent 3 details

if_ref #VisParent3 is_not(*null)

Change #FP_EKEY3 #VisParent3.uUserObjectType

endif

* Object details

if_ref #uObject is_not(*null)

#DF_ELTXTL := #uType + ' ' + #uObject.uCaption + ' ' + #uObject.uIdentifier

+ ' ' + #uObject.uUserObjectType

endif

endroutine

End_Com

UF_QRY03 - Listening for Ramp Objects And View Their Properties

```
*
=====
*
* Component   : UF_QRY03
* Type       : Reusable Component
* Ancestor   : VF_AC010 (Command Handler)
*
* PLEASE NOTE: This UF_ (User Framework) component is the shipped
version. You
* may choose to modify it. You should do this by copying the source
* code of this component into your own component and then change
* the copied version. This will prevent the accidental loss of your
* changes if you upgrade your Visual LANSA framework version. Refer
* to the end of this component for more details about making your
* own version of this component.
*
* This is example code only - No warranty is expressed or implied.
* Neither this program, nor any derivative of it, should be ever be used in
* production or end-user environments.
*
*
=====
*
* This demonstrates how to listen for RAMP objects and view their properties
*
=====
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #VF_AC010) Height(336)
Layoutmanager(#ATLM_2) Width(552)
*
=====
* Simple Field and Group Definitions
*
=====
*
```

=====
* Component definitions

*

=====
* Body and Button arrangement panels

Define_Com Class(#PRIM_PANL) Name(#BUTTON_PANEL)
Displayposition(2) Height(311) Layoutmanager(#BUTTON_FLOW)
Left(464) Parent(#COM_OWNER) Tabposition(2) Tabstop(False) Top(25)
Width(88)

Define_Com Class(#PRIM_PANL) Name(#BODY_HEAD)
Displayposition(1) Height(311) Layoutmanager(#ATLM_1) Left(0)
Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(25)
Verticalscroll(True) Width(464)

* Attachment and flow layout managers

Define_Com Class(#PRIM_ATLM) Name(#MAIN_LAYOUT)
Define_Com Class(#PRIM_ATLI) Name(#BUTTON_ATTACH)
Attachment(Right) Manage(#BUTTON_PANEL) Parent(#MAIN_LAYOUT)
Define_Com Class(#PRIM_FWLM) Name(#BUTTON_FLOW)
Direction(TopToBottom) Flowoperation(Center) Marginbottom(4)
Marginleft(4) Marginright(4) Margintop(4) Spacing(4) Spacingitems(4)

Define_Com Class(#PRIM_ATLI) Name(#BODY_ATTACH)
Attachment(Center) Manage(#BODY_HEAD) Parent(#MAIN_LAYOUT)

* The report button

Define_Com Class(#PRIM_PHBN) Name(#PHBN_REPORT)
Buttondefault(True) Caption('Report') Displayposition(1) Left(4)
Parent(#BUTTON_PANEL) Tabposition(1) Top(4)
Define_Com Class(#PRIM_FWLI) Name(#FWLI_SAVE_BUTTON)
Manage(#PHBN_REPORT) Parent(#BUTTON_FLOW)

* The Traverse component

Define_Com Class(#UF_TRVRS) Name(#UF_TRVRS)

* The Output List

Define_Com Class(#PRIM_GRID) Name(#LTVW_1) Displayposition(1)
Height(142) Left(2) Parent(#BODY_HEAD) Rowresize(True)
Showselection(True) Showsortarrow(True) Tabposition(1) Top(2) Width(460)
Define_Com Class(#PRIM_GDCL) Name(#LVCL_4) Displayposition(1)
Parent(#LTVW_1) Source(#DF_ELXTL) Width(100)
Widthtype(Remainder)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1) Marginbottom(2)
Marginleft(2) Marginright(2) Margintop(2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center)
Parent(#ATLM_1)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_2) Attachment(Center)
Manage(#BODY_HEAD) Parent(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_3) Attachment(Right)
Manage(#BUTTON_PANEL) Parent(#ATLM_2)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_4) Attachment(Center)
Manage(#LTVW_1) Parent(#ATLM_1)

* The Script List

Define_Com Class(#PRIM_MEMO) Name(#SCRIPT) Componentversion(1)
Currentline(1) Displayposition(2) Height(165) Left(2)
Maximumlinelength(200) Parent(#BODY_HEAD)
Showselectionhighlight(False) Tabposition(2) Top(144) Width(460)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_5) Attachment(Bottom)
Manage(#SCRIPT) Parent(#ATLM_1)
Define_Com Class(#PRIM_MECL) Name(#MECL_1) Columnrole(Data)
Displayposition(1) Parent(#SCRIPT) Source(#VF_ELXTB)
Define_Com Class(#PRIM_GDCL) Name(#GDCL_1) Caption('Member ID')
Captiontype(Caption) Displayposition(2) Parent(#LTVW_1)
Source(#VF_ELIDN)

* Show Script push button

Define_Com Class(#PRIM_PHBN) Name(#PHBN_1) Displayposition(3)
Left(0) Parent(#COM_OWNER) Tabposition(3) Top(0) Width(552)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_6) Attachment(Top)
Manage(#PHBN_1) Parent(#ATLM_2)

Define_Com Class(#PRIM_PHBN) Name(#PHBN_SCRIPT) Caption('Show Script') Displayposition(2) Left(4) Parent(#BUTTON_PANEL) Tabposition(2) Top(33)

Define_Com Class(#PRIM_FWLI) Name(#FWLI_1)
Manage(#PHBN_SCRIPT) Parent(#BUTTON_FLOW)

Define_Com Class(#PRIM_PHBN) Name(#PHBN_FKEY) Caption('Show FKeys') Displayposition(3) Left(4) Parent(#BUTTON_PANEL) Tabposition(3) Top(62)

Define_Com Class(#PRIM_FWLI) Name(#FWLI_2)
Manage(#PHBN_FKEY) Parent(#BUTTON_FLOW)

* Framework - class #VF_FP001 #TempVF_FP001 Reference(*Dynamic)

* Application - class #VF_FP002 #TempVF_FP002 Reference(*Dynamic)

* Business Object - class #VF_FP003 #TempVF_FP003

Reference(*Dynamic)

* Command - class #VF_FPM09 #TempVF_FPM09 Reference(*Dynamic)

* Command Usage - class #VF_FPM10 #TempVF_FPM10

Reference(*Dynamic)

* Filter - class #VF_FPM14 #TempVF_FPM14 Reference(*Dynamic)

*

=====
* Events Definitions

*

=====
*

=====
* Property Definitions

*

=====
*

=====
* Method Definitions

*

=====
* -----

* Handle Command Execution

```

* -----

Mthroutine Name(uExecute) Options(*REDEFINE)

* The return code field and testing condition

Define #Ret_Code reffld(#IO$STS)
Def_cond Name(*RetOkay) Cond('#Ret_Code = OK')

* Do any execution logic defined in the ancestor

Invoke #Com_Ancessor.uExecute

Endroutine

*
=====
* Subroutines
*
=====
*
=====
* Event Handlers
*
=====

* -----
* Handle the save button
* -----

EVTROUTINE HANDLING(#PHBN_REPORT.Click)

* Tell the Traverse reusable part to read through the entire system in no
particular order
* This program listens for the signals for each new object,

clr_list #LTVW_1

```

```
invoke #UF_TRVRS.uTraverseFramework
```

```
ENDROUTINE
```

```
* Listen for the RAMP Container
```

```
EVTROUTINE HANDLING(#UF_TRVRS.RAMPContainer)
```

```
Reference(#TempVF_FP025)
```

```
* Output generic details
```

```
Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP Session')  
uObject(#TempVF_FP025)
```

```
* For more details of the RAMP Session object, click on #TempVF_FP025  
above and press F2, and look at the properties of class VF_FP025
```

```
add_entry #LTVW_1
```

```
* Store a reference to the object against the list line
```

```
Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FP025)
```

```
ENDROUTINE
```

```
* Listen for a RAMP Session
```

```
EVTROUTINE HANDLING(#UF_TRVRS.RAMPSession)
```

```
Reference(#TempVF_FP026)
```

```
* Output generic details
```

```
Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP Group')  
uObject(#TempVF_FP026)
```

```
* For more details of the RAMP Group object, click on #TempVF_FP026  
above and press F2, and look at the properties of class VF_FP026
```

```
add_entry #LTVW_1
```

* Store a reference to the object against the list line

```
Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FP026)
```

```
ENDROUTINE
```

* Listen for a RAMP Destination

```
EVTROUTINE HANDLING(#UF_TRVRS.RAMPDestination)  
Reference(#TempVF_FPM27)
```

* Output generic details

```
Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP  
Destination')
```

* Specific details

* For more details of the RAMP Destination object, click on

#TempVF_FPM27 above and press F2, and look at the properties of class
VF_FPM27

* Store the member Identifier for this destination

```
Change #VF_ELIDN #TempVF_FPM27.CurrentMemberGUID
```

```
#DF_ELXTL := 'Destination ' + #TempVF_FPM27.uCaption + ' Technical  
Caption ' + #TempVF_FPM27.uTechnicalCaption + ' Navigate Script ID ' +  
#TempVF_FPM27.uNavigateScriptIDN + ' Return Script ID ' +  
#TempVF_FPM27.uReturnScriptIDN
```

```
add_entry #LTVW_1
```

* Store a reference to the object against the list line

```
Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FPM27)
```

```
ENDROUTINE
```

* Listen for a RAMP Junction

```
EVTROUTINE HANDLING(#UF_TRVRS.RAMPJunction)
Reference(#TempVF_FPM28)
```

* Output generic details

```
Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP Destination')
```

* Specific details

* For more details of the RAMP Destination object, click on #TempVF_FPM28 above and press F2, and look at the properties of class VF_FPM28

* Store the member Identifier for this junction

```
Change #VF_ELIDN #TempVF_FPM28.CurrentMemberGUID
```

```
#DF_ELTXTL := 'Junction ' + #TempVF_FPM28.uCaption + ' Technical
Caption ' + #TempVF_FPM28.uTechnicalCaption + ' Script ID ' +
#TempVF_FPM28.uIdentifier + ' Screen Name ' +
#TempVF_FPM28.uScreenName
```

```
add_entry #LTVW_1
```

```
ENDROUTINE
```

* Listen for a RAMP Special

```
EVTROUTINE HANDLING(#UF_TRVRS.RAMPSpecial)
Reference(#TempVF_FPM29)
```

* Output generic details

```
Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP Special')
```

* Specific details

* For more details of the RAMP Destination object, click on #TempVF_FPM29 above and press F2, and look at the properties of class

VF_FPM29

* Store the member Identifier for this special

Change #VF_ELIDN #TempVF_FPM29.CurrentMemberGUID

#DF_ELTXTL := 'Special ' + #TempVF_FPM29.uCaption + ' Technical
Caption ' + #TempVF_FPM29.uTechnicalCaption

add_entry #LTVW_1

ENDROUTINE

* Listen for a RAMP Script

EVTROUTINE HANDLING(#UF_TRVRS.RAMPScript)

Reference(#TempVF_FPM30)

* Output generic details

Invoke #Com_Owner.uOutputGeneric uLevel(1) uType('RAMP Script')

* Specific details

* For more details of the RAMP Destination object, click on

#TempVF_FPM30 above and press F2, and look at the properties of class
VF_FPM30

* Store the member Identifier for this script

Change #VF_ELIDN #TempVF_FPM30.CurrentMemberGUID

#DF_ELTXTL := 'Script ' + #TempVF_FPM30.uCaption + ' Technical
Caption ' + #TempVF_FPM30.uTechnicalCaption

add_entry #LTVW_1

* Store a reference to the object against the list line

Set Com(#ltvw_1.currentitem) Relatedreference(#TempVF_FPM30)

ENDROUTINE

* Output a line of description of the object

```
mthroutine uOutputGeneric
define_map *input #std_num #uLevel
define_map *input #df_elmsg #uType
Define_Map For(*input) Class(#vf_fp001) Name(#VisParent1)
Mandatory(*NULL) Pass(*BY_REFERENCE)
Define_Map For(*input) Class(#vf_fp002) Name(#VisParent2)
Mandatory(*NULL) Pass(*BY_REFERENCE)
Define_Map For(*input) Class(#vf_fp003) Name(#VisParent3)
Mandatory(*NULL) Pass(*BY_REFERENCE)
Define_Map For(*input) Class(#vf_ac001) Name(#uObject)
Mandatory(*NULL) Pass(*BY_REFERENCE)
```

Change #vf_elidn *blanks

* Object details

```
if_ref #uObject is_not(*null)
#DF_ELXTL := #uType + ' ' + #uObject.uCaption + ' ' + #uObject.uIdentifier
+ ' ' + #uObject.uUserObjectType
endif
```

endroutine

* Button press - Show the lines of script for a script

EVTROUTINE HANDLING(#PHBN_SCRIPT.Click)

```
Define_Com Class(#VF_FPM30) Name(#TempVF_FPM30)
Reference(*DYNAMIC)
```

```
selectlist #LTVW_1
```

```
if #LTVW_1.CurrentItem.Selected
```

```
If_Ref Com(#LTVW_1.CurrentItem.RelatedReference) Is(*INSTANCE_OF
#VF_FPM30)
```

```
set_ref #TempVF_FPM30 (*Dynamic
#LTVW_1.CurrentItem.RelatedReference)
invoke #Com_Owner.uShowScript uScriptManager(#TempVF_FPM30)
UseScriptGUID(#VF_ELIDN)

endif
endif
endselect
```

```
ENDROUTINE
```

```
* Show a script
```

```
mthroutine uShowScript
Define_Map For(*input) Class(#VF_FPM30) Name(#uScriptManager)
Pass(*BY_REFERENCE)
define_map *input #VF_ELMBRi #UseScriptGUID
```

```
define #First reffld(#vf_elbool)
Define_Com Class(#vf_elindx) Name(#LoopLimit)
```

```
clr_list #Script
```

```
* When working with any VF_FPM objects you have to set the
CurrentMemberGUID before reading or writing to them.
```

```
* This is stored in field #VF_ELIDN in list #LTVW_1
```

```
* set #VF_FPM30 to the correct member
#uScriptManager.CurrentMemberGUID := #UseScriptGUID
```

```
Change Field(#LoopLimit) To(#uScriptManager.uSLMax)
Change Field(#VF_ELLICT) To(0)
```

```
Change #First TRUE
Begin_Loop To(#LoopLimit)
* Read each script line into field #vf_eltxtb
```

```
Invoke Method(#uSystem.uRestorePropertySet) Ucursor('0') Ufirst(#First)
Uname(uSL) Ualphavalue(#vf_eltxtb) Udefaultalphavalue('*NONE')
Ac0x1object(#uScriptManager.SetMember( #UseScriptGUID ))
```

```
Add_Entry To_List(#SCRIPT)
```

```
Change #First FALSE
```

```
End_Loop
```

```
endroutine
```

```
* Button press - Show the function keys for a destination
```

```
EVTROUTINE HANDLING(#PHBN_FKEY.Click)
```

```
Define_Com Class(#VF_FPM27) Name(#TempVF_FPM27)
```

```
Reference(*DYNAMIC)
```

```
selectlist #LTVW_1
```

```
if #LTVW_1.CurrentItem.Selected
```

```
If_Ref Com(#LTVW_1.CurrentItem.RelatedReference) Is(*INSTANCE_OF
#VF_FPM27)
```

```
set_ref #TempVF_FPM27 (*Dynamic
```

```
#LTVW_1.CurrentItem.RelatedReference)
```

```
invoke #Com_Owner.uShowFKeys uDestinationManager(#TempVF_FPM27)
```

```
UseDestGUID(#VF_ELIDN)
```

```
endif
```

```
endif
```

```
endselect
```

```
ENDROUTINE
```

```
* Show the function keys for a destination
```

```
Mthroutine Name(uShowFkeys)
```

```
Define_Map For(*input) Class(#VF_FPM27) Name(#uDestinationManager)
```

```

Pass(*BY_REFERENCE)
define_map *input #VF_ELMBRi #UseDestGUID

Define_Com Class(#vf_elindx) Name(#LoopLimit)
Define_Com Class(#vf_elindx) Name(#LoopIndex)
Define_Com Class(#vf_elindx) Name(#KeyTotal)

Clr_List Named(#SCRIPT)

* When working with any VF_FPM objects you have to set the
CurrentMemberGUID before reading or writing to them.
* This is stored in field #VF_ELIDN in list #LTVW_1

Set_Ref Com(#uDestinationManager ) To(#uFramework.VF_FP027Manager)
#uDestinationManager.CurrentMemberGUID := #UseDestGUID
Change Field(#LoopLimit) To(#uDestinationManager.uKeyTotal)

* Load cherry function keys

Begin_Loop Using(#LoopIndex) To(#LoopLimit)

Change Field(#vf_elindx) To(#LoopIndex)

#vf_eltxtb := ' Caption: ' +
#uDestinationManager.uKeyCaption<#LoopIndex>

#vf_eltxtb += ' , '

#vf_eltxtb += ' Enabled in Newlook: ' +
#uDestinationManager.uKeyEnabledNL<#LoopIndex>

#vf_eltxtb += ' , '

#vf_eltxtb += ' Enabled in VLF: ' +
#uDestinationManager.uKeyEnabledVLF<#LoopIndex>

#vf_eltxtb += ' , '

```

```
#vf_eltxtb += ' Key to Send: ' +  
#uDestinationManager.uKeytoSend<#LoopIndex>
```

```
Add_Entry To_List(#SCRIPT)
```

```
End_Loop  
endroutine
```

```
End_Com
```

Troubleshooting

[Troubleshooting Filters and Command Handlers](#)

[Troubleshooting Snap-In Instance Lists](#)

[Debugging Browser Problems](#)

Troubleshooting Filters and Command Handlers

Framework fails as soon as you select your new Windows filter or command handler

When you recompile your Windows filter or command handler it fails with a "Permission Denied" error.

Your browser application filter or command handler behaves unexpectedly or erratically

Framework fails as soon as you select your new Windows filter or command handler

Environment: Windows

Solution: There is something terminally wrong with your new filter or command handler. You should check all of the following:

- Your filter or command handler is a Reusable Part.
- Your filter has the correct ancestor (VF_AC007 for filters, VF_AC010 for command handlers).
- Your filter or command handler does not have an RDMLX level coding error (e.g.: divide by zero) that causes it to fail when starting. Carefully check all of the messages issued for the exact cause of the error. Try running your filter or command handler in Visual LANSAs debug mode.

When you recompile your Windows filter or command handler it fails with a "Permission Denied" error.

Environment: Windows

Solution: Your filter or command handler is active while you are trying to compile it. Shut down the Visual LANSA Framework while recompiling active filters or command handlers.

Your browser application filter or command handler behaves unexpectedly or erratically

Environment: WAM

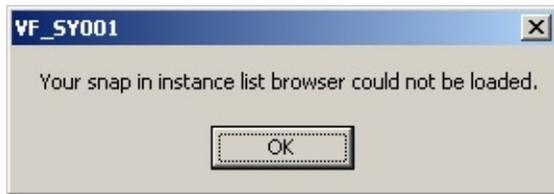
Solution: Work through this checklist first:

- Initially try executing your Framework with tracing mode on. This may reveal that your filter is not processing events the way that you think it should be.

If the problem persists try adding specific trace commands to your function that indicate in more detail what it is doing and what data it is processing. Check in and recompile your function, then execute your Framework with tracing mode turned on. Continue to add trace statements and recompile your function until the logic problem is located.

Troubleshooting Snap-In Instance Lists

Problem: You receive a message indicating that your snap-in instance list could not be loaded: **Environment:** Windows



Solution: There is something terminally wrong with your new snap-in instance list.

You should check all of the following:

- Your snap-in browser is a reusable part.
- Your snap-in browser has ancestor VF_AC012.
- Your snap-in browser does not have an RDMLX level coding error (e.g.: divide by zero) that causes it to fail when starting. Carefully check all of the messages issued for the exact cause of the error. Try running your snap-in browser in debug mode.

Problem: When you recompile your snap-in browser the compile fails with a "Permission Denied" error. **Environment:** Windows

Solution: Your snap-in browser is active while you are trying to compile it. Shut down the Framework while recompiling snap-in browsers.

Debugging Browser Problems

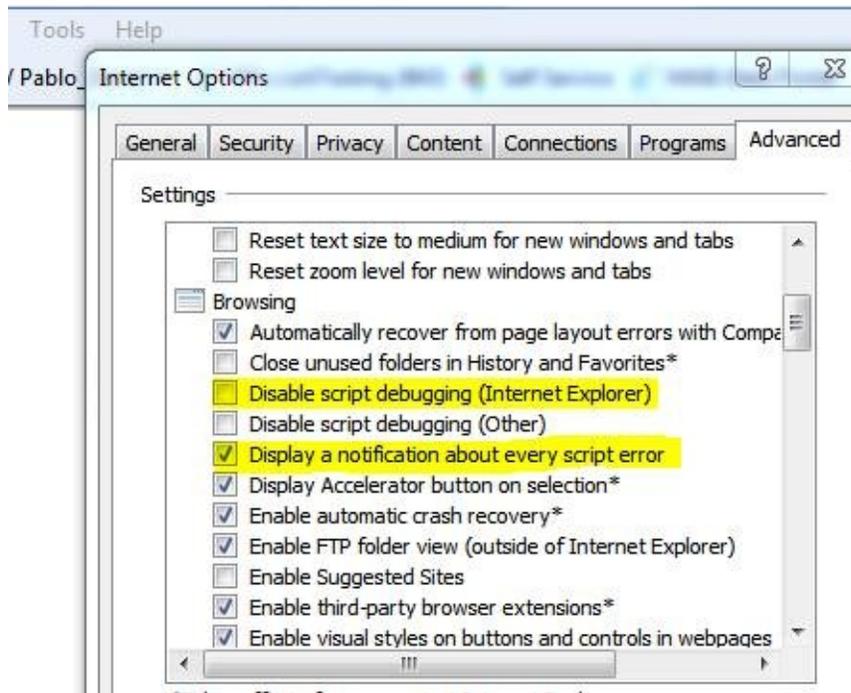
Internet Explorer

Firefox, Chrome and Safari

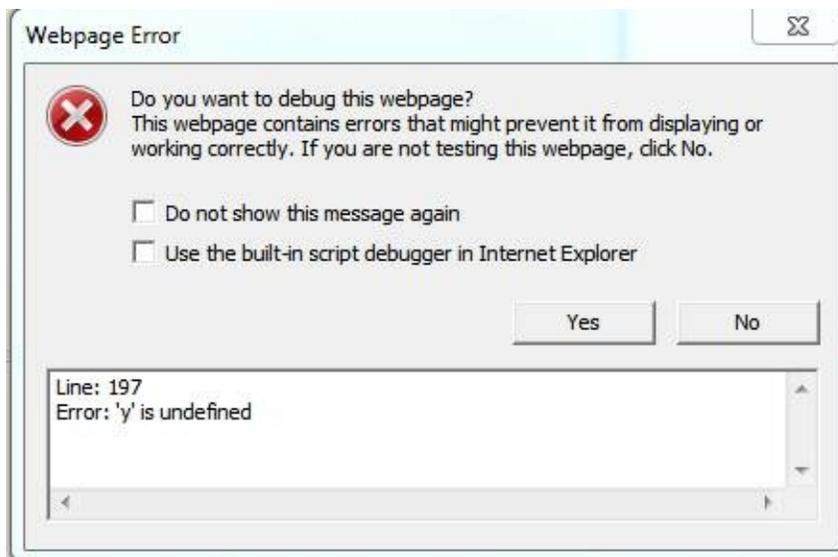
Internet Explorer

Make sure under the Tools menu, Internet Options, Advanced tab:

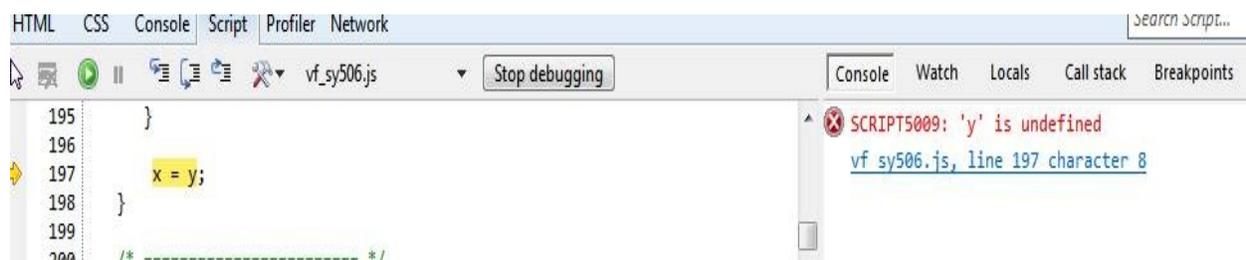
- Disable script debugging is unchecked.
- Display a notification about every script error is checked.



With those settings when there is a javascript error a window like this should appear:



If you check the box Use the built-in script debugger in Internet Explorer a window with error highlighted should come, something like this:



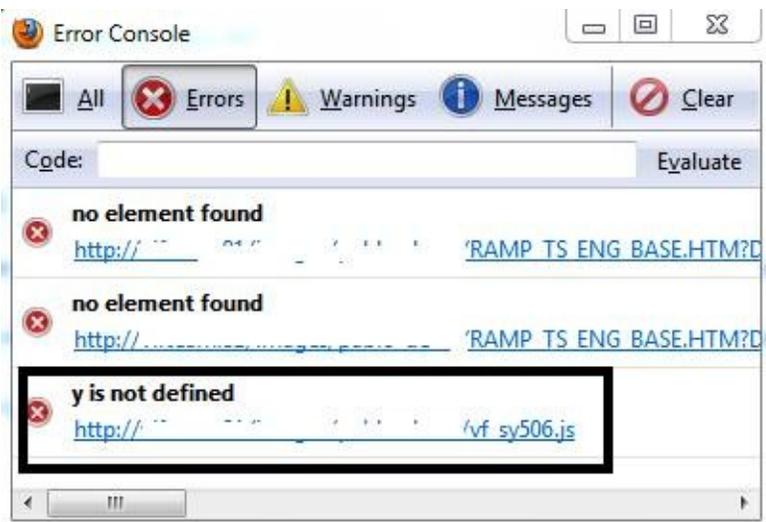
Firefox, Chrome and Safari

Unlike IE, the errors will not show up but they will be recorded into their respective web consoles.

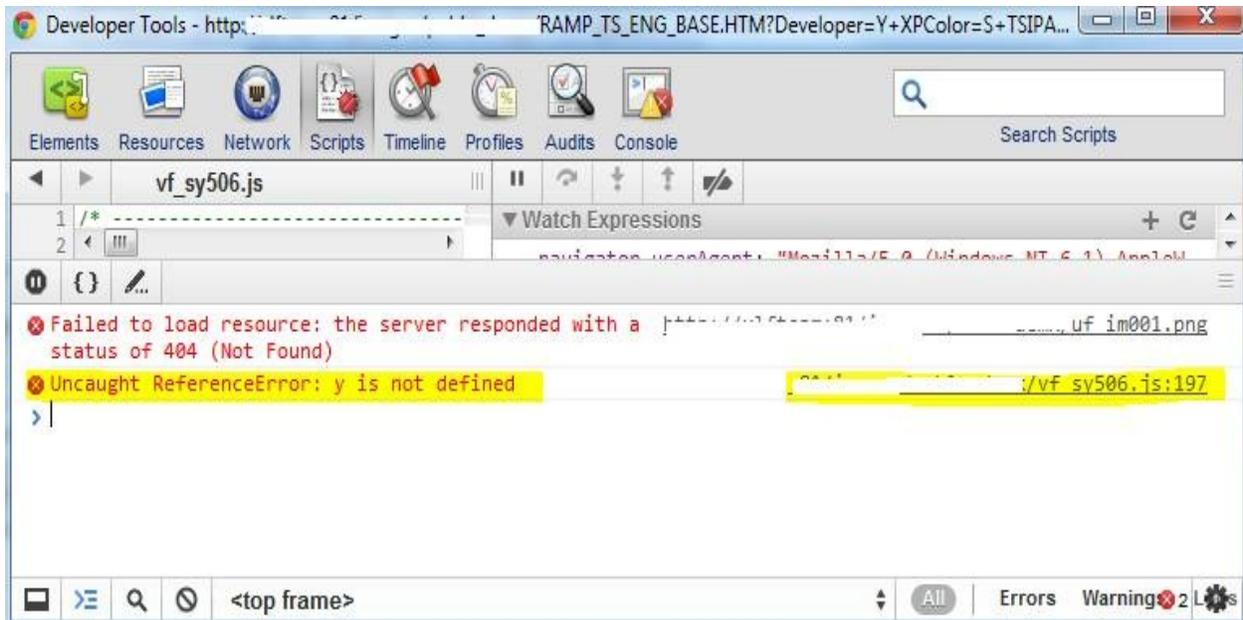
When the browser seems not to be responsive it is probably because some sort of error occurred.

Firefox and Chrome: press Ctrl+Shift+J

Firefox Error Console:



Chrome error console:



Safari: press Ctrl+Alt+C



Whichever browser you are using the information above will be very useful for technical support purposes.

Application Performance

Framework based applications may be deployed as Windows Client-Server applications and/or as Web browser applications.

The nature of the technologies you will be using to accomplish this mean that there are a large number of risk variables that may affect the performance of your application (this actually has little to do with the Framework or with LANSA specifically).

Some risk variable are under your direct control and are affected by what you do during the design phase of your project. Others may not be under your direct control. Some may only appear when you deploy your application. Generally you should identify and manage risks as early as possible in your project.

These sections describe some of the risk variables you may be dealing with and some of the ways that you might manage them:

[Work as an End-User](#)

[Best-Case Conditions](#)

[Worst-Case Conditions](#)

[A Risk Check List](#)

[Assessing Performance in Framework Web Applications](#)

[Using CITRIX, ISA and Windows Terminal Server Projected Desktop Technologies](#)

[Options for Very Large Frameworks](#)

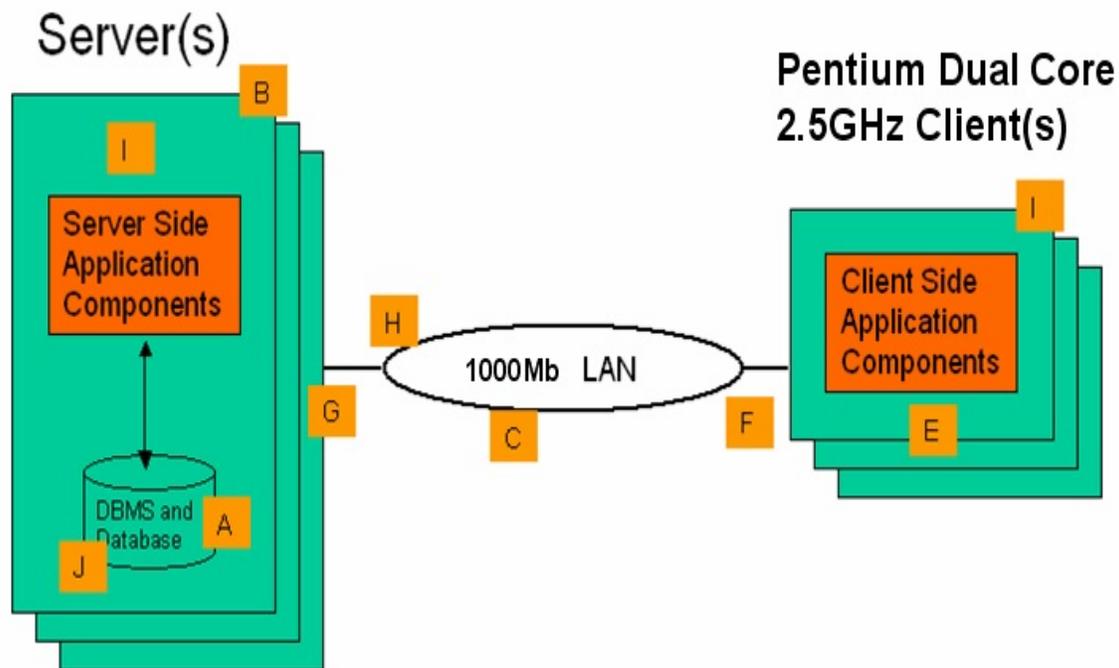
Work as an End-User

All performance testing should be performed by using the Framework as either an administrator or an end-user.

Using the Framework as a designer produces a markedly different performance profile.

Best-Case Conditions

For many Windows or Web browser applications this type of operating environment represents best-case conditions:



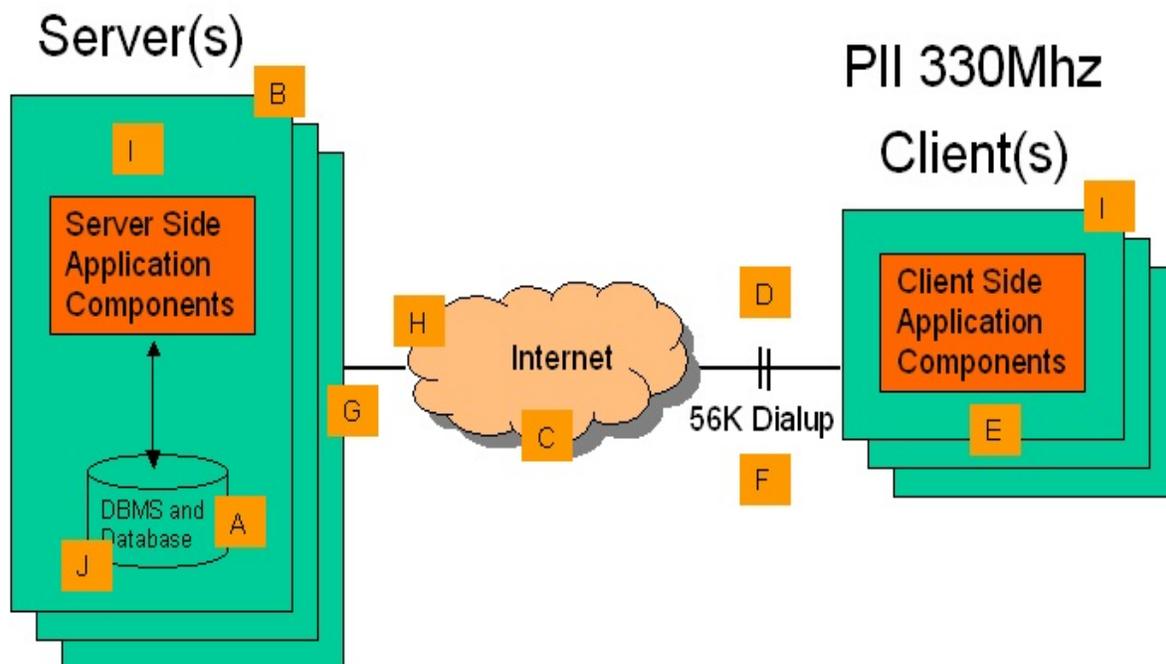
Even when you are deploying to best-case conditions there are a number of performance risks that you face.

It is important when designing and implementing any application to identify **your** best case and worst case operating conditions as early as possible and then assess the risks that exist in the gap between them.

See [A Risk Check List](#) for help in identifying and assessing some of your possible risks.

Worst-Case Conditions

For many Windows or Web browser applications this type of operating environment represents worst-case conditions:



When you are deploying to worst-case conditions there are a number of very significant performance risks that you face.

It is important when designing and implementing any application to identify **your** best case and worst case operating conditions as early as possible and to then assess the risks that exist in the gap between them.

See [A Risk Check List](#) for help in identifying and assessing some of your possible risks.

A Risk Check List

General Risks

Risk	Possible Issues	Possible Mitigation
Develop and test under best case conditions and deploy to worst case conditions	Poor performance Applications over-functioned.	Set up a worst case environment and use it early and frequently during development to identify and eliminate problem areas prior to deployment.
Develop and test under worst case conditions and deploy to best case conditions	Poor developer productivity Applications under-functioned.	Provide developers with better operating conditions

Specific Risks (column 1 is the letter shown in the [Best-Case Conditions](#) & [Worst-Case Conditions](#) diagrams).

Risk	Description	Possible Issues / Comments	Possible Mitigation
A	Database volume and load not reflective of deployed conditions		Load development databases with realistic volumes as soon as possible during development. Use load emulation tools during development and testing.
B	Server cannot support number of clients	Cost overruns Brick wall to future growth	Hardware upgrade. Use load emulation tools during development and testing.

C	Variable throughput of Internet connections, even with apparently high bandwidths.	Peak time poor throughput Lack of control over ISPs and uncontrolled communication paths	Expectation management
D	Too much data being transferred		Reduce data to realistic volumes.
E	Client logic too complex		Reduce complexity of client logic to match power of client systems.
F	Client accessing server too often		Reduce number of times that communications links are crossed per transaction.
G	Firewall, proxy, wireless and VPN configuration differences in deployed environments	Configuration issues Installation issues Performance variations Reliability variations Unanticipated costs	
H	LAN load	Unanticipated costs Brick wall to future growth	
I	OS or Application Server differences in deployed environments	Configuration issues Installation issues Performance	

		variations Reliability variations Unanticipated costs	
J	DBMS differences in deployed environments	Configuration issues Installation issues Performance variations Reliability variations Unanticipated costs	

Assessing Performance in Framework Web Applications

To resolve a performance issue in your Framework web application you need to do this repeatedly:

1. Identify the **most significant** problem area.
2. Make a change that you think will alleviate the problem.
3. Assess the impact of the change (is it actually better or worse?)
4. Identify the next most significant problem area and go back to step 2.

You need to use a structured and reproducible technique in identifying the problem area and assessing any changes you make. Following are some suggestions about doing this.

Step 1. Use consistent and reproducible environments

Step 2. Make a Reproducible Assessment Script (ie: a RAS)

Step 3. Perform your RAS - To get a Baseline or to Test a Change

Step 4. Assessing the Average CPU per Server Interaction

Step 5. Identify and Assess any Client-side Problem Areas

Step 6. Identify and Assess any Server Side Problem Areas

Step 1. Use consistent and reproducible environments

There are some obvious things you need to do.

- Use the same:
 - PC
 - web server
 - application server
 - communications channel and method
- Perform exactly the same actions on each test.
- Make your test like a real end-user's experience in order to better identify significant improvements you can make.
- Use a PC that is configured like an 'average' end-user's PC.

It would be ideal if your web server, application server and communications channel were always under the same load. Practically this is rarely the case, so you need to be able to compensate for this. For example performing a test at 6:00am, in an empty office, and the comparing the elapsed times with another test taken at 10:30am, when everyone is working flat out, would be silly.

Normally you can make sure your PC is under the same load, but sometimes something as simple as receiving a large e-mail during a test can invalidate your results.

Define exactly what your PC, web server and application server will be on paper for later reference.

Step 2. Make a Reproducible Assessment Script (ie: a RAS)

- Choose the application areas that the most end-users use most of the time. In many commercial applications 10% of the application is often used 90% of the time by 90% of the users. Always assess the 10% that is used most first, which is what most people are most likely to complain about.
- Choose the application area which the most business benefit will be gained from any improvements. This probably means using the 90/10 rule again, but even within that, improving just one area like Order Entry might be 95% of the overall business benefit gain.
- Create a RAS **that involves 30 or more interactions** with the web server.
- In the RAS record exactly how many server interactions are performed.
- Rather than just cruising around randomly in the application, make the RAS work as end-users would work, completing typical actions, especially where cyclical activities are repeatedly performed.
- Write the RAS down as a table of scripted steps so that it can be easily repeated. This also allows other people to be delegated to complete the test cycles for you.
- Include a 'Perceived Elapsed Response Time' (PERT) column in which the perceived response time of each step can be recorded by using a stopwatch.
- Producing and keeping a RAS for each test you perform is useful as it documents any gradual improvements you make. It also provides, in a very subjective and emotive area, something objective that can be used as the basis for discussions with management, end-users, etc.

Step 3. Perform your RAS - To get a Baseline or to Test a Change

- Start the VLF application in your PC browser and move around in it. Use the STATS=Y option. Do not use developer=Y. It is important in applications that are used for long periods of time to “warm them up” before commencing a test. Delays that are experienced the first time that something is used when it is going to be used many times anyway, will just confuse your results and make them atypical of what a real end-users experience most of the time. This may lead you to wrong conclusions about what is the most significant improvement you can make.
- Locate the L4WEB job on the server that is servicing your browser.
- Get positioned at step 1 of your RAS.
- Record the current CPU time for your L4Web job on the server.
- Perform the RAS exactly
- Record the perceived elapsed response time (PERT) for each step.
- Save the STATS=Y information in the VLF window.
- Check the CPU time of your L4Web on the server and record the delta (ie: how much CPU time was used on the server to complete execution of your RAS).
- Divide the CPU time used by the number of interactions in your script to get an average CPU time per server interaction (this is why not performing the script exactly each time will ruin your results).

Step 4. Assessing the Average CPU per Server Interaction

What is your average server CPU time per server interaction from your RAS?

If the average number is greater than 1.0, you need to stop and think carefully about your application and your performance expectations. Even if this number is lower (0.3 or less) and you are going to have a lot of application users, you need to think carefully about this number and what it means.

You should do some rough calculations here about average CPU cycles per transaction, the number of concurrent users, user clicking rates / server hit rates, etc, and see if what you want to achieve and what it is possible to ever achieve are at least in the same ballpark.

Note: A Ballpark Example – Imagine providing 1000 users, who all hit the server on average every 5 seconds, in an application with a 1.5 average CPU seconds per transaction time. They expect a 2.0 second PERT time. In a 5 second interval your server needs to provide (even just theoretically and grossly simplistically) at least $1000 \times 1.5 = 1500$ CPU seconds. That's asking it provide 300 CPU seconds of processing power per elapsed second, which is nonsensical. Maybe 32 processors would help? It would, but you'd still probably be out by at least an order of magnitude.

Performance tuning your System i server will not normally reduce this number (it appears that sometimes it will if you are using complex SQL requests).

Normally IBM i level performance tuning will only impact the elapsed time of an application.

You can only really reduce this number by:

- Performance tuning your programs to use less CPU cycles.
- Increasing the power of your System i server so that it can execute more CPU cycles per second.

Imagine a high average CPU time of 2.0 per interaction.

This means your application requires 2.0 seconds of dedicated CPU cycles to complete its work (on average). This absolutely does not mean you can expect a 2.0, 3.0 or even 5.0 second PERT time unless it's 11:15pm at night, you are on a 1000Mb LAN, using a memory resident application, and that nobody else is using the server or communications network.

On a busy system, getting access to 2.0 seconds of CPU cycles may take 30 elapsed seconds (or even a lot more) as your L4Web job competes with other L4Web jobs, 5250 sessions, batch jobs, the Apache HTTP server, network file

managers, printer writers, etc.

Step 5. Identify and Assess any Client-side Problem Areas

CLIENT-SIDE			
Executed	Avg (secs)	Min (secs)	Max (secs)

Examine the parts of the STATS=Y report that relate to your RAS.

The client-side numbers shown here are elapsed times. They indicate how long the JavaScript in your VLF client took to execute. This means the numbers are impacted by other things that are happening on your PC. If you receive an e-mail, or if MS-Word, SVCHOST.EXE or RUNDLL32.EXE are in mysterious loops using 50% of your PC's CPU cycles, then the numbers shown here will be impacted.

The Executed Column

Look at all the entries with Executed = 1.

Something that is only executed once in a RAS, when a typical end-user actually uses in many times, is a suspect number and may lead you to fail to identify the most significant performance improvement areas. In this case, rewrite your RAS so that the executed number is more reflective of an end-user experience.

The Avg Column

Where the executed value is reflective of an end-user's experience, look at the average (Avg) value. Is it higher than 1.0? If it is, then this is probably indicating that what you are trying to do in your client-side JavaScript is too complex for the PC being used for the RAS.

Some areas to look at changing to assess their impacts include reducing the number of:

- Lists being displayed
- Entries in the instance list
- Use a more powerful PC

The Difference Between STATS=Y and PERT Times

The Avg time displayed is an elapsed time. This means the Avg number and your perceived elapsed response time (PERT) should be similar, but will probably not be the same.

Your PERT time includes the time it takes IE to 'draw' the HTML document onto the screen.

If the Avg value and the associated PERT averages consistently vary by more than 1 second you should investigate further. One potential cause of problems in this area is presenting very large instance lists on underpowered PCs. The version of IE being used may also impact this area. It is reported that IE7 is better at screen drawing than IE6, and that IE7 under Vista is even better still.

Step 6. Identify and Assess any Server Side Problem Areas

SERVER-SIDE			
Executed	Avg (secs)	Min (secs)	Max (secs)

The numbers shown here are the elapsed time from when the request was sent to the server until a response was received back. If a number is higher than you think it should be and you want to reduce it there are several things you need to look at:

Your WAM programs are too complex for the System i server

This should show up when you do step ‘Step 4 – Assessing the Average CPU per server interaction’. We have all done this with 5250 programs, batch programs, etc. The programs are just too complex for the computer to process when a lot of people start to use them at the same time. Web programs are no different. The solutions to this type of problem are the same as for 5250 programs or batch programs.

Note – Using a ‘Hello World’ baseline You may be able to classify this type of problem by using a simple ‘Hello World’ program in your VLF application test harness. If the ‘Hello World’ program runs well and your other program runs poorly, then the most likely cause of the problem is that the other program is overly complex.

Your communications system and/or HTTP server are overloaded or poorly configured

If you have a reasonable average CPU time in your L4Web interactions, it may be that the time taken to get the HTTP request to your server (or back again) is too long and needs to be improved.

Overloaded or Slow Communication Routes

Assessing this is an area that requires specialized tools and resources. However, if you are working in an environment where everybody knows that from 9:00am until 11:30am printing something on the local printer goes very slowly, and that copying a small file to the network sever takes 30 seconds, then you almost certainly have an issue in this area that will impact your L4Web jobs.

Apache Server Tuning

You Apache web server schedules the execution of your HTTP requests. It may be overloaded or performing poorly for many reasons. For example, it may not be configured properly and not able to use enough execution threads to handle

the number of requests being sent to it. Equally, threads may be being blocked or lost to failing requests.

- Make sure you all the very latest Apache PTFs applied.
- Review the IBM supplied Redbooks related to Apache tuning
- Consult an expert in Apache performance tuning.
- **Note** – Using a ‘Hello World’ baseline - You may be able to classify this type of problem by using a simple ‘Hello World’ program in your VLF application test harness. If the ‘Hello World’ program and your other program both run poorly, then you may have HTTP server or communication issues.

Your L4Web jobs are taking too long to execute

If you have a reasonable average CPU time in your L4Web interactions, it may that your L4Web job is taking too long to processes the request. For example, it might require 0.3 CPU seconds to complete, but be taking 40 elapsed seconds to get access to the 0.4 seconds of CPU time. This is exactly the same type of problem that can impact 5250 programs or batch programs. The slow response time happens because the program can’t get access to CPU cycles.

Investigating this is vanilla System i tuning stuff ... subsystems, machine pool storage activity levels, time-slices, priorities, machine pool sizes, etc. Broadly speaking, you would be looking for the same sort of signs of trouble as in a 5250 application, most typically exhibited in very high paging rates in the pools in which the L4Web jobs execute.

This sort of assessment may culminate in a hardware upgrade because it turns out to be cheaper than the man-hours spent trying to force too much work into too small a computer.

You may be having this trouble in this area for these reasons:

- When you plan to introduce 50 new 5250 users to you System i you would naturally consider the impact this would have on its daily workload, possibly having to resize the machine as a result. Did you do this for your L4Web users?
- If you did, did you account for the fact that a web user typically uses more resources than a 5250 user? For example, a web user in an AJAX-style application like the VLF will typically have very short ‘think times’ and be hitting the System i CPU for cycles much more often that a 5250 user with their much longer ‘think times’.

Same things to try here are:

- Make your programs simpler so that they match the capabilities of the server.
- Restructure your application to reduce the frequency of 'hits' on your server.
- Performance tune your System i server, especially if very high pool paging rates are observed.
- Put more memory onto your System i server to reduce pool paging rates.
- Upgrade your System i Server

Step 7. Identifying the Next Most Significant Problem Area

After making your assessments, identify the single thing that you think will produce the most significant improvement.

Make the required changes and repeat steps 3 through 6 again using the RAS.

If it makes things worse, as they sometimes do, back the change out.

If it makes things better, identify and make the next most significant improvement, then repeat steps 3 through 7 again.

Using CITRIX, ISA and Windows Terminal Server Projected Desktop Technologies

If your application is to be deployed via a projected desktop technology such as CITRIX or Windows Terminal Server, you need to ensure that you test early and often in an environment that will reflect your ultimate deployment environment. For example, 100 users concurrently starting MS-Office on their individual desktops presents a significant load on their individual desktop systems but little load on the server other than basic file serving (and only then if they are sharing MS-Office resources on the server in some way). However 100 users starting MS-Office in a projected desktop environment reverses this load. The server (where the 100 instances of MS-Office are really being started concurrently) is placed under a significant load while the clients experience only a light load in projecting the MS-Office desktop images.

Projected desktop technologies are attractive propositions from the maintenance, control and deployment points of view, but you need to ensure that your server(s) are capable of sustaining the aggregate client load. If you design an application that is predicated on having significant desktop CPU power and resources available on each individual client system, then change your deployment model to use a projected desktop technology, you will need to ensure that your server(s) can handle the sum total of all the client's concurrent CPU loads and resource requirements.

Options for Very Large Frameworks

Some Frameworks may contain more than 500 business objects. With frameworks this large the start up times, both for users and developers, can become annoying. Additionally having multiple developers working in with a single XML framework definition file may present management issues.

An alternative to a large framework is to subdivide it into multiple smaller frameworks. Assuming you can make the subdivision (on an application or role-orientated basis probably), you could be dealing with for example three frameworks defined like this:

Entry Point Form Name Framework XML Definition File

MJDFRAME1 Frame_1.XML

MJDFRAME2 Frame_2.XML

MJDFRAME3 Frame_3.XML

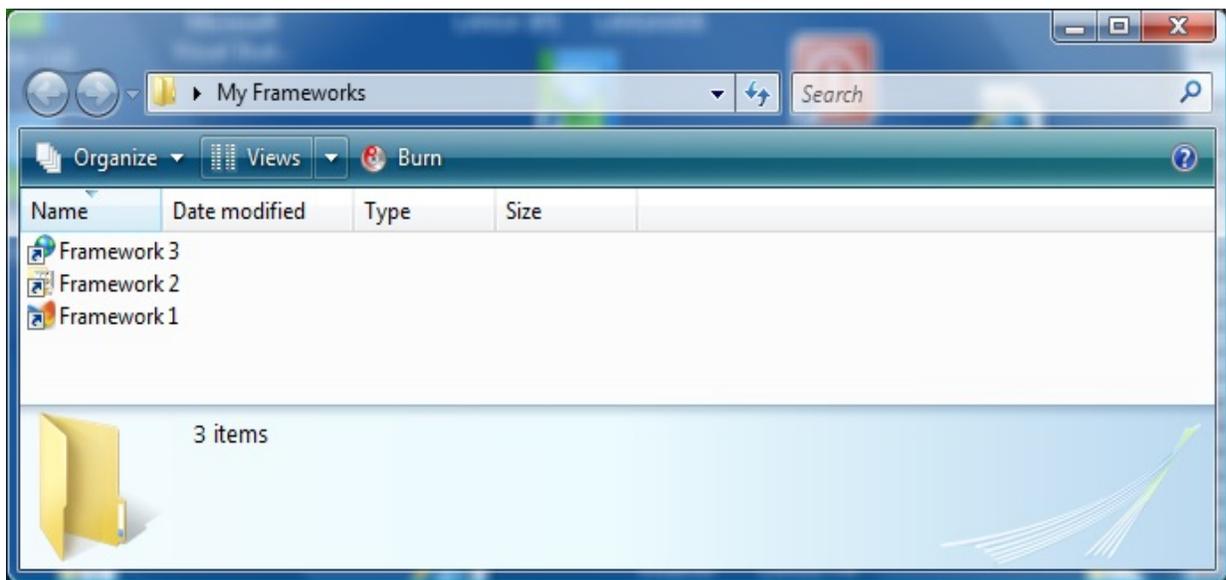
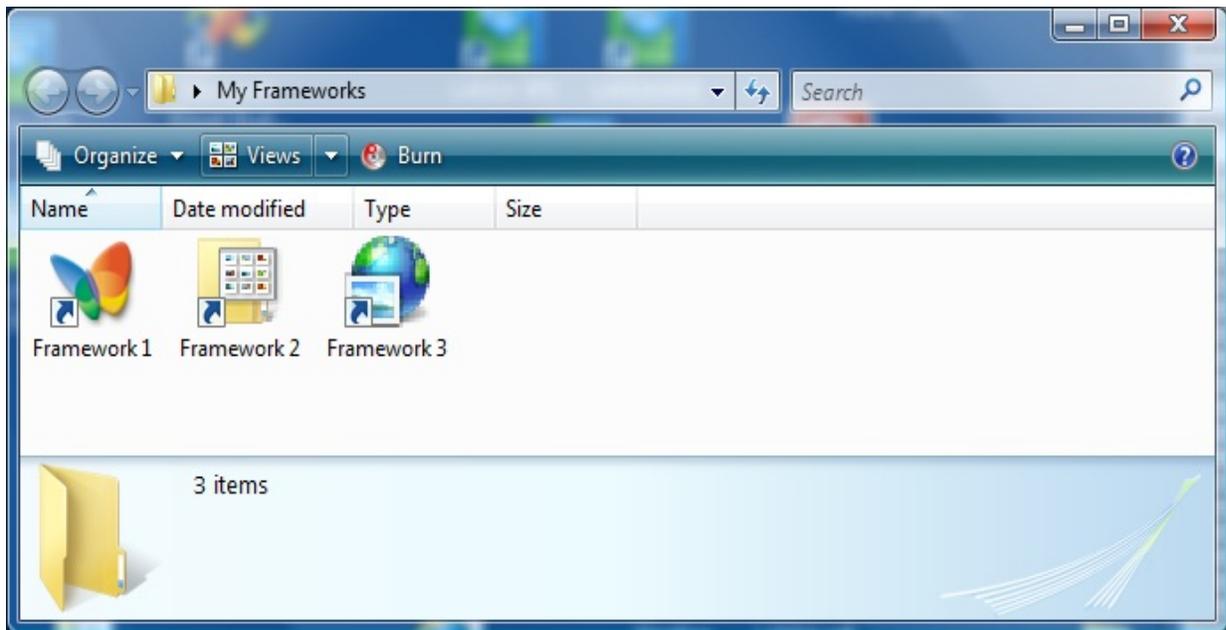
The three entry point forms would just be normal VLF entry point forms:

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #VF_AC006)

Mthroutine Name(uInitializeFramework) Options(*Redefine)
  Set Com(#Com_Owner) IDesignMode(FALSE) Uadminmode(FALSE)
  Set Com(#Com_Owner) Usystemxmlfile('<<Framework's XML file
definition file>>')
Endroutine

End_com
```

Once you have three frameworks you can put them in a folder on the user desktop and use it like a high level menu:



This menu system offers a high level alternative to the main VLF navigation tree. It also allows end-users a large choice in the way they arrange and display things.

Also, it is usually easier to swap between multiple application areas open in different windows than by moving around in the VLF navigation tree.

You can also fairly easily allow one framework to start up another. This VLF hidden command handler will generically launch another VL form:

```
BEGIN_COM ROLE(*EXTENDS #VF_AC020)
```

```
MTHROUTINE NAME(uExecute) OPTIONS(*REDEFINE)
```

```
Use OV_SYSTEM_SERVICE With_Args(START_LANSA ('FORM=' +  
#Com_Owner.avAlphaArg1))
```

```
Endroutine
```

```
End_com
```

Let's say Framework 1 contains applications called Framework 2, Framework 3 and Other Frameworks. The Other Frameworks application also contains business objects titled Framework 2 and Framework 3. In Framework 1 the navigation pane looks like this:

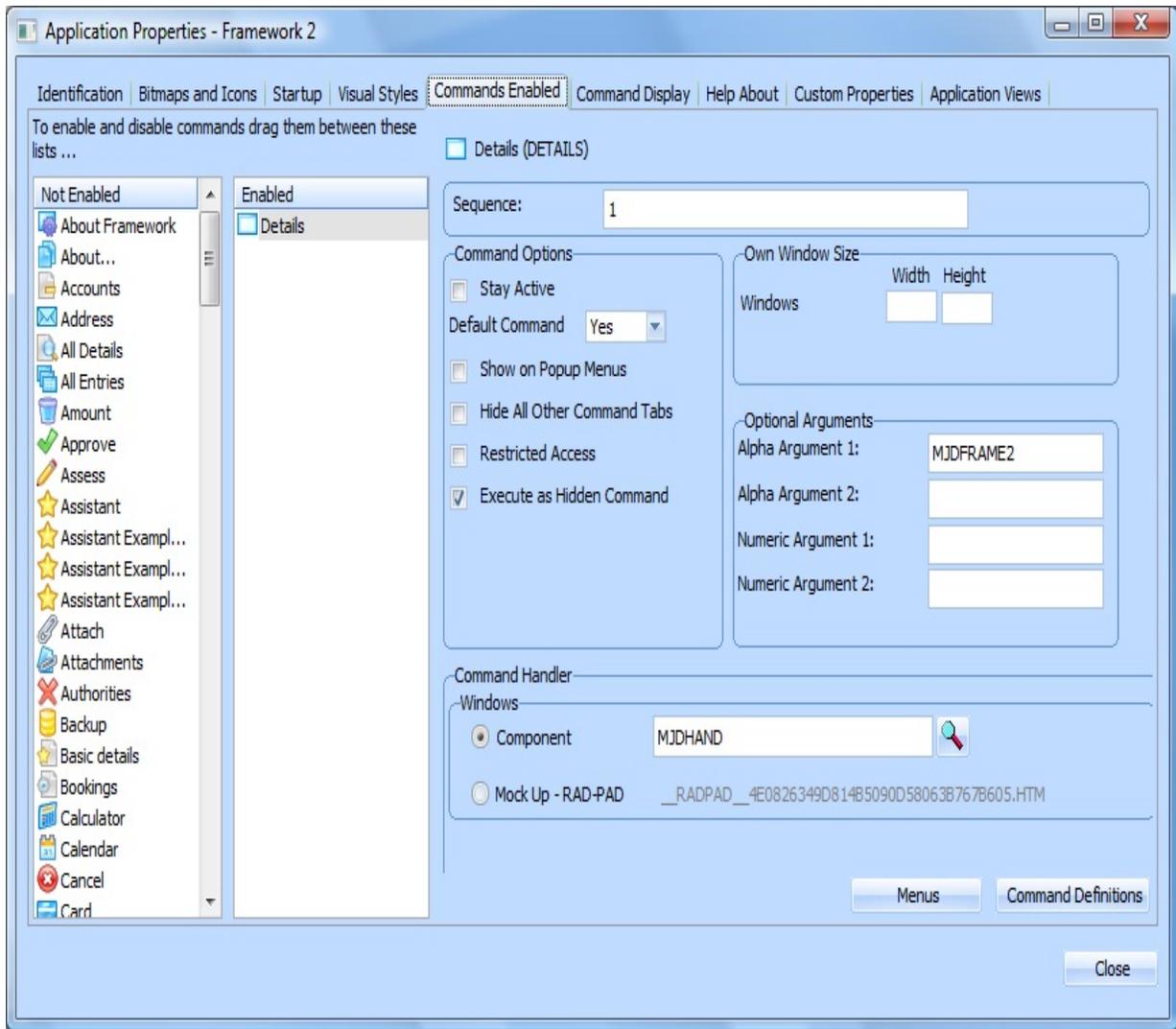


This shows two styles of accessing Framework 2 or 3 from within Framework 1.

If you click on Framework 2 or Framework 3 at any level in the Framework 1 navigation tree, the other framework is launched as another VL process by the generic command handler above.

If Framework 2 contained 300 business objects (say), you have now made them accessible from within Framework 1 without having to bear the overhead of actually having them defined in Framework 1. All you had to do was to add to Framework 1 a single application or business object to act as a link.

The same command handler would be defined for each application and business object, like this:



The salient points are that the command handler is defined as a default command, it does not show on popup menus, it executes as hidden command, and the alpha argument 1 contains the name of the VL form to be launched by the command handler.

You can now start your frameworks from individual desktop icons or from the navigation tree in Framework 1.

There are some additions to this example that are relatively easy to make. You can use:

- A marker file or a data queue to check whether the other framework is already active and avoid starting up another instance.
- Your own logon IIP and an encrypted temporary stream file to pass logon details between frameworks. That way Framework 1 can capture the last set of logon/connection details. When framework 2 or 3 start up they can

bypass asking the user to supply the logon details again.

- Data queues to communicate between different frameworks. If each framework has a 'framework manager' that listens to an associated queue, it is relatively easy to communicate between the frameworks.

Using techniques like this you can produce a result that is just as integrated and probably easier to use than a single very large framework.

Definitions

This section describes the properties and options in the Framework.

The easiest way to view the definitions is to use the [Assistant](#) when running the Framework. Select the object you want information for and press F2. The definition of the object is show in the Assistant window.

[Framework Window](#)

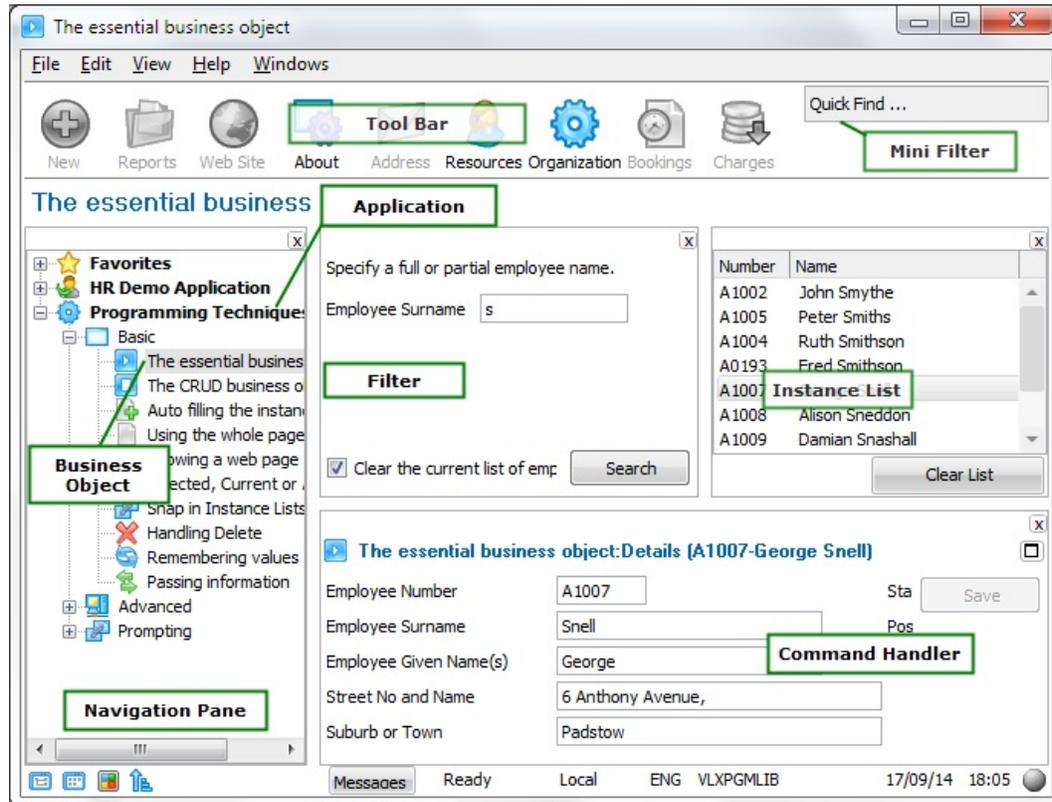
[Property Tab Sheets](#)

[Dialogs](#)

[Properties](#)

Framework Window

- Title Bar
 - Toolbar
 - Menu Bar
 - Popup Menu
 - Navigation
Pane
 - Application
Bar
-
- Business Object
Bar
 - Instance
List
 - Command
Tab
 - Application
View



Title Bar

[Framework Window](#)

The horizontal bar below the toolbar in which the name of the currently selected [Application](#) or [Business Object](#) is displayed.

The command handler folder also has a title bar showing the business object to which the displayed commands apply.

Toolbar

[Framework Window](#)

The toolbar has buttons for commonly used commands.

Menu Bar

[Framework Window](#)

The menu bar contains menus in which commands are grouped.

It is recommended that all commands in the Framework are included in the menus on the menu bar.

Popup Menu

[Framework Window](#)

A context sensitive menu for the most commonly used commands for the selected object.

Navigation Pane

A vertical panel positioned by default on the left of the window where applications and business objects are displayed.

Application Bar

[Framework Window](#)

The left-hand-side of the navigation bar in which [Applications](#) are displayed.

Application and Business Object Tree

[Framework Window](#)

The application and business object tree is an alternative way of displaying the [Application Bar](#) and the [Business Object Bar](#).

Business Object Bar

[Framework Window](#)

A panel the next to the application bar in which [Business Objects](#) are displayed.

Application View

[Framework Window](#)

Intermediate groupings of business objects. You can have up to ten application views of business objects in an application.

Filter Tab

[Framework Window](#)

A tab showing the [Filter](#) for a business object.

Filter Folder

[Framework Window](#)

A folder containing tabs for all the filters for a business objects.

By default the tab is shown between the business object bar and the instance list.

Instance List

[Framework Window](#)

A list showing all the selected business object instances, in other words items of a specified business object type.

Filters control which instances are displayed.

See also:

[Framework Programming](#)

[Filters and List Manager](#)

[Command Handlers and List Manager](#)

[More about Instance Lists](#)

Command Tab

[Framework Window](#)

A tab showing the command handlers for a [Command](#).

Command Folder

[Framework Window](#)

A folder containing tabs for all the [Commands](#) applicable for the selected application, business object or instance.

By default the folder is displayed on the bottom of the Framework window.

Property Tab Sheets

Application Views

Authorities

Bitmaps and Icons

Commands Enabled

Command Display

Custom Properties

Developer Preferences – Web Server

Export Design

Filter Snap-in Settings

Filter Settings

Framework Details

Help About

Icons

Identification

Instance List/Relations

Other Options

Server Details

Startup

Toolbar and Menus

Usage

User Administration Settings

User Details

Visual Styles

Visualization

Web/RAMP Details

Application Views

Use this tab to specify application views for the currently selected application. You can define up to 10 views for each application.

Application Properties - HR Demo Application

Identification Bitmaps and Icons Startup Visual Styles Commands Enabled Command Display Help About Custom Properties Application Views

Bs. Objects not in Views

Identification Bitmaps and Icons Create Links

Caption Bs. Objects not in Views (ENG)

Hint:

Sequence: 2

Internal Identifier: 05E64D31D73A41B78A26708E5481E439

Unique Identifier: 49

User Object Name / Type 05E64D31D73A41B78A26708E5481E439

Restricted Access

Allow on Web

Allow in Windows

Allow this Object to be Opened in a New Window Manually

Last Changed 20140921-155001-VLXPGMLIB

Alter Development Status

New Delete

Close

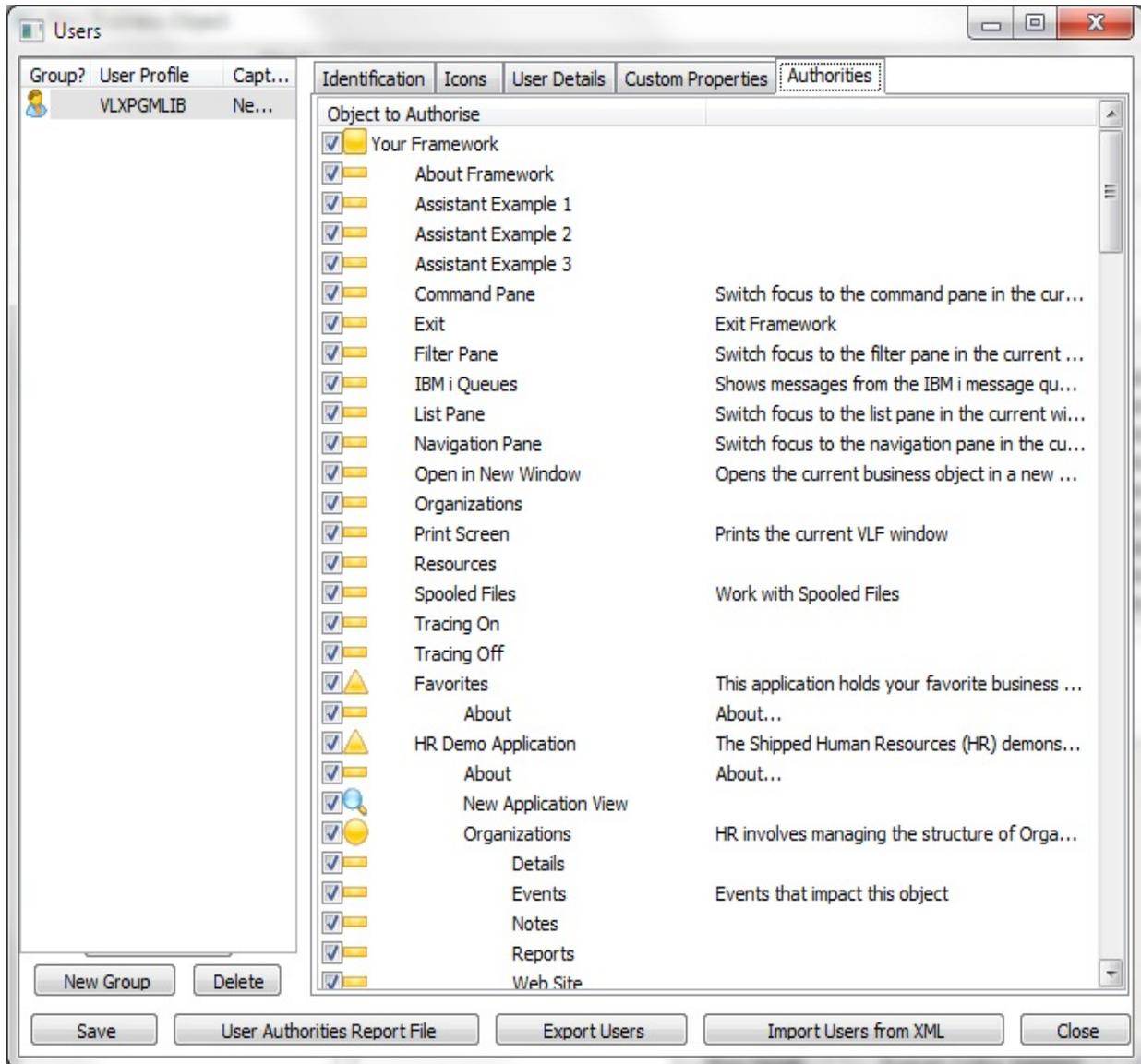
Identification

[Bitmaps and Icons](#)

[Create Links](#)

Authorities

Use this tab to set authorities to Framework objects for the selected user.



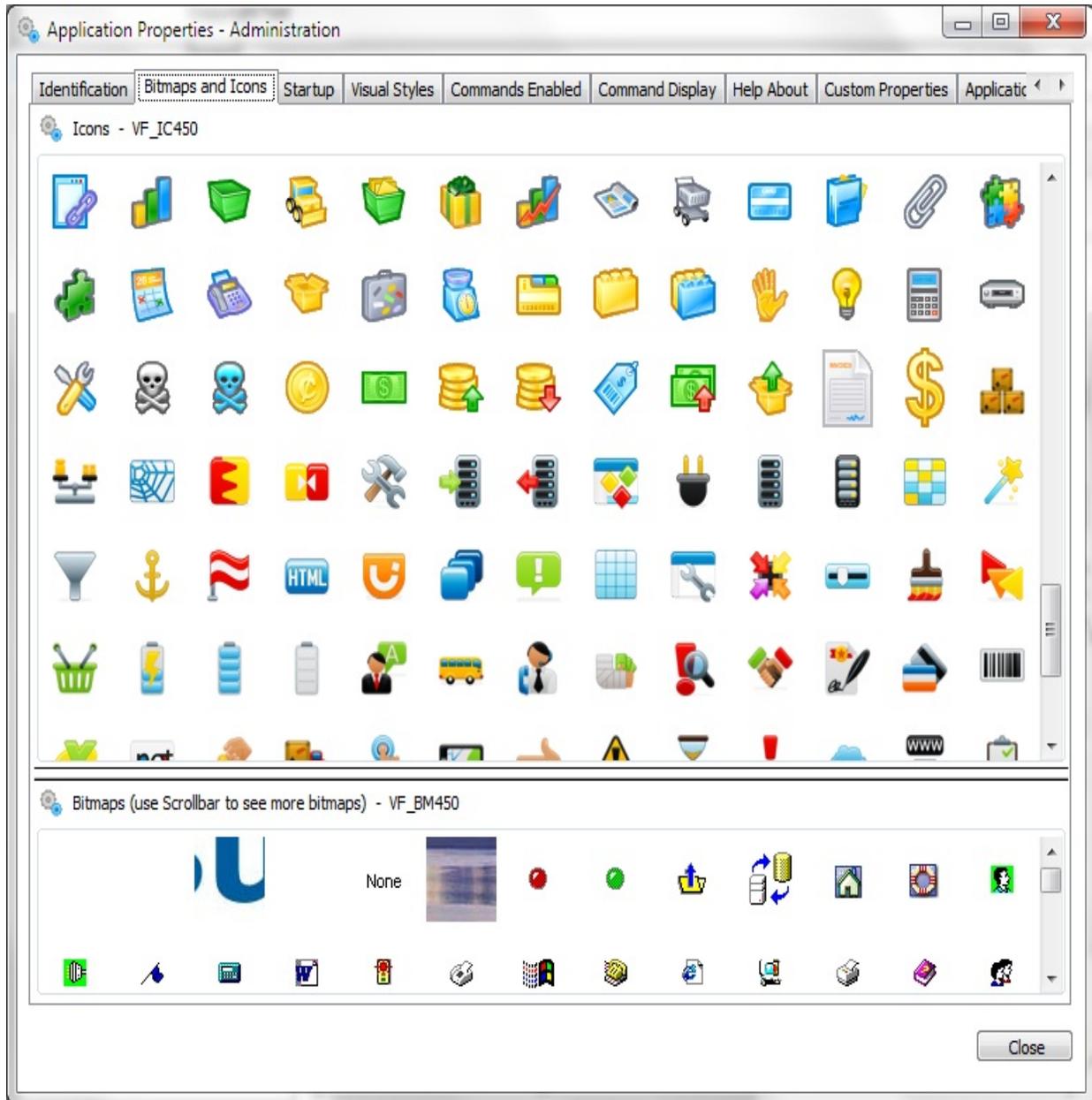
[User Authorities Report File](#)

[Export Users](#)

[Import Users from XML](#)

Bitmaps and Icons

Use this tab to set icons and bitmaps for an object:

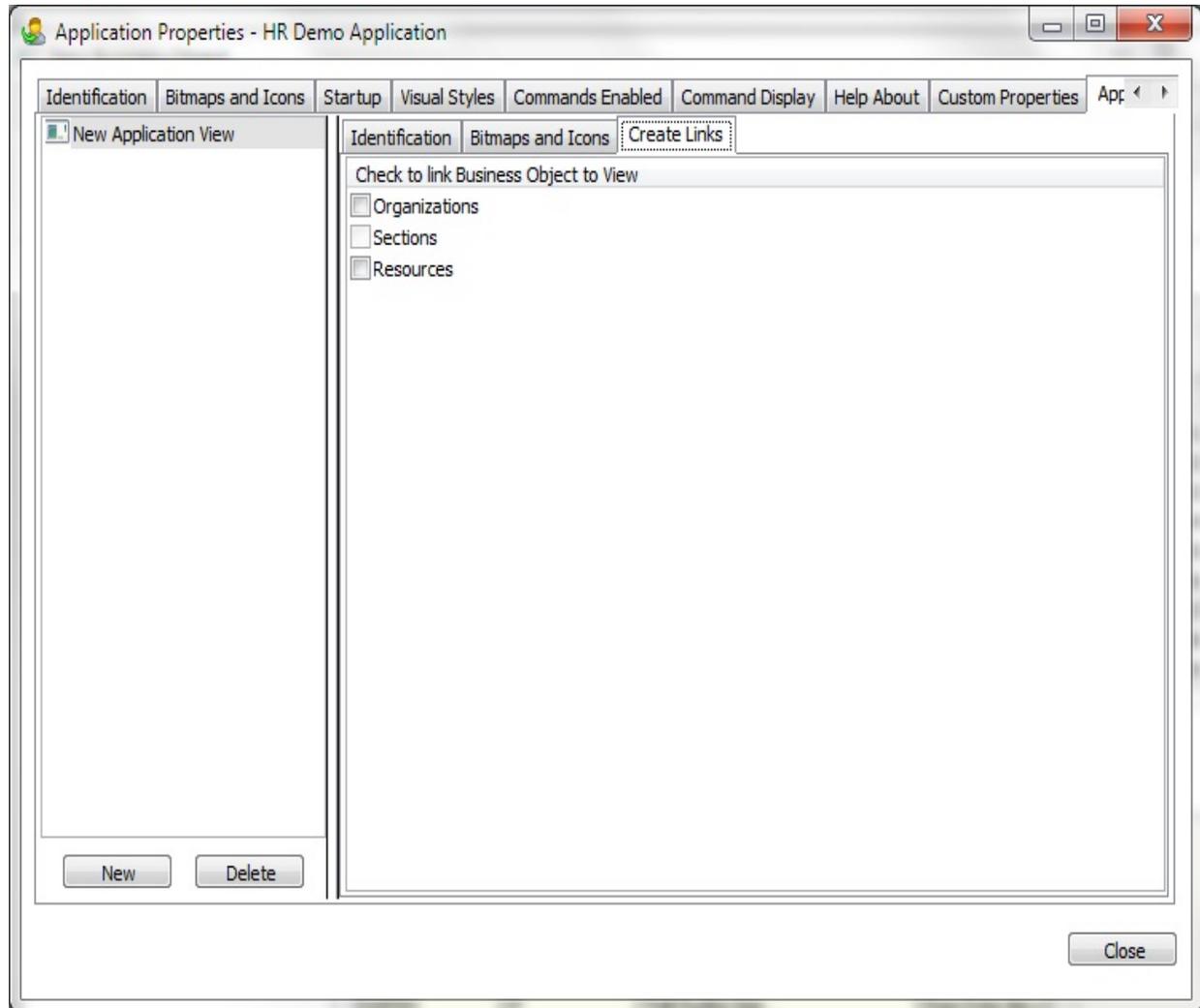


Icon

Bitmap

Create Links

This list shows all the business objects currently defined for the selected application.



To link a business object to a view, first select the view you want to link the business object to and then check the box beside the business object.

Commands Enabled

Use this tab to specify which commands are enabled for your application or business object.

Business Object Properties - Resources

Identification | Icons | Visual Styles | Filters | Filter Settings | **Commands Enabled** | Command Display | Custom Properties | SubTypes | Instance List / Relations

To enable and disable commands drag them between these lists ...

Not Enabled

- About
- About Fram...
- Accounts
- Address
- All Details
- All Entries
- Amount
- Approve
- Assess
- Assistant
- Assistant Ex...
- Assistant Ex...
- Assistant Ex...
- Attach
- Attachments
- Authorities
- Backup
- Basic details
- Bookings
- Calculator
- Calendar
- Cancel
- Card
- Category
- Charges
- Checks
- Cheques
- Claims
- Close
- Command P...
- Connect
- Contacts
- Contents
- Copy
- Costs
- CRUD
- Custom Pro...
- Cut
- Dates
- Delete
- Delete profile
- Delivery Ad...
- Departments

Enabled

- Details
- Documents
- Events
- Images
- Notes
- Reports
- TimeSheets

Details (DETAILS)

Choose Command Type

Business Object Command Instance Command

Sequence:

Command Options

Stay Active:

Default Command:

Allow on Web

Allow in Windows

Show on Popup Menus

Show on Instance List Tool Bar

Hide All Other Command Tabs

Restricted Access

Execute as Hidden Command

Bypass Locks

Own Window Size

	Width	Height
VLF-WIN - Windows	<input type="text"/>	<input type="text"/>
VLF-WEB / VLF.NET - Web Browser	<input type="text"/>	<input type="text"/>

Optional Arguments

Alpha Argument 1:

Alpha Argument 2:

Numeric Argument 1:

Numeric Argument 2:

Alter Development Status ▾

Command Handler

VLF-WIN - Windows

Component Identifier 🔍

Mock Up - RAD-PAD

VLF-WEB / VLF.NET - Web Browser

WEBEVENT/Hidden Function (Id) 🔍 Process: VF_PR003

Location for Buttons:

Type of Layout Style to be Used:

WAM Identifier 🔍

AJAX HTML Page Module 🔍

Mock Up - RAD-PAD

.NET Component

Class Name: 🔍

Assembly:

RAMP Destination(s) - Use RAMP Tools to change

*WEBEVENT Function and/or WAM Component or AJAX Routine

AJAX Page (HTML File)

Allow in Windows

Allow on Web

Alter Development Status

Associated AJAX Function

Business Object Command

Bypass Locks

Component Identifier

Default Command

Enable Command

Execute as Hidden Command

Hide All Other Command Tabs

Instance Command

Location for Buttons

Mock Up RAD-PAD

.NET Component Class Name and Assembly

Show on Instance List Tool Bar

Show On Popup Menus

Optional Arguments

Own Window Size

RAMP Destinations

Restricted Access

Sequence

Stay Active

Type of Layout Style to be Used

See also:

Command

Command Handler

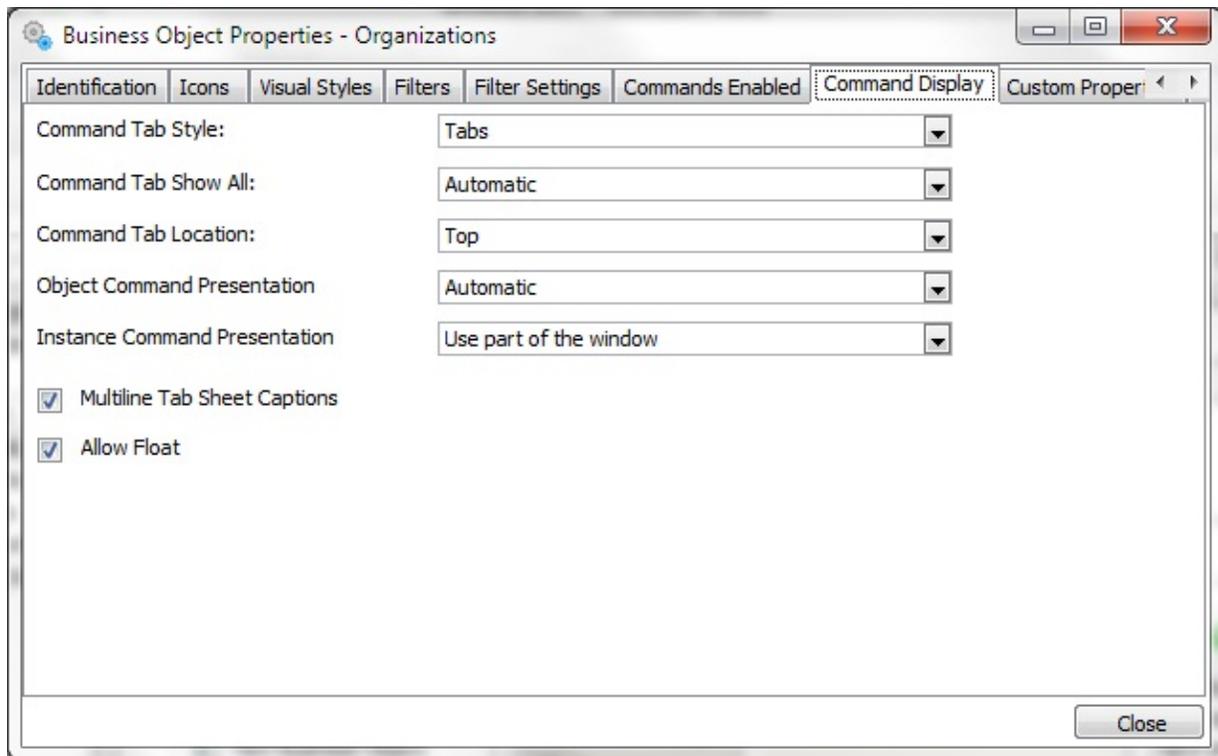
Mock Up Filters and Command Handlers

Images Palette

Prototype Your Commands and Their Handlers
Create Your Own Command Handlers

Command Display

Use this tab to define how commands are displayed:



Note that not all properties are available for all object types.

[Allow Float](#)

[Command Tab Location](#)

[Command Tab Show All](#)

[Command Tab Style](#)

[Instance Command Presentation](#)

[Multiline Tab Sheet Captions](#)

[Object Command Presentation](#)

Custom Properties

Use this tab to define custom properties:

Se...	Caption	Name
1	New User Pro...	USER_PROPERTY

Defined In: Your Framework

Caption: New User Property

Sequence: 1

Name: USER_PROPERTY

Help Text:

Property Type:

- Alphanumeric
- Numeric
- Boolean

Maximum Length: 1

Maximum Decimals: 0

Uppercase

Input Method:

- Single Value
- List of Values
- Select from a Fixed List

Maximum Entries in List: 1

Allow Multiple Selection

Value(s) can be changed by Administrator

Fixed/Default Values:

[Allow Multiple Selection](#)

[Caption](#)

[Defined In](#)

[Fixed / Default Values](#)

[Help Text](#)

[Input Method](#)

[Maximum Decimals](#)

[Maximum Entries in List](#)

[Maximum Length](#)

[Name](#)

[Property Type](#)

[Sequence](#)

[Uppercase](#)

[Value\(s\) can be changed by Administrator](#)

See also:

[Custom Properties](#)

[Frequently Asked Questions about Custom Properties](#)

[Things to be careful with when using Custom Properties](#)

Read Only

Code Table Definition/ Use a Reusable Part

Code Table Definition/ Reusable Part Data Handler (ID)

Developer Preferences – Web Server

Use this tab to set up the preferences for each of the web servers you are using (maximum 2):

The screenshot shows a Windows-style dialog box titled "Framework" with a tabbed interface. The active tab is "Developer Preferences - Web Server 1". The dialog is organized into several sections:

- Development and Testing Details:**
 - Web Server Caption: Not Available (with a blue "(ENG)" link)
 - Preferred web scheme/skin: Windows Classic (dropdown menu)
- Details of the LANSAs for the Web server you will use while doing development:**
 - Host Name or IP Address: [text box] with a "Verify Web" button
 - Images Folder: [text box] with a "Verify" button
 - Private Working Folder: [text box] with a "Verify" button
- Optional Mapped Drives to your LANSAs for the Web Folders:**
 - Images Folder: [text box] with a "Verify" button
 - Private Working Folder: [text box] with a "Verify" button
- Script for Uploading to your LANSAs for the Web Folders:**
 - Script Type: FTP File Transfer Commands (dropdown menu) with a "Generate Example" button
 - [Large empty text area for script content]
- .NET Compilation Options:**
 - Certificate File (PFX): [text box] with a "Verify" button
 - Certificate File Password: [text box]

A "Close" button is located at the bottom right of the dialog.

Certificate File (PFX)

Certificate File Password

Host Name or IP Address

Images Folder

[Optional Mapped Drives - Images Folder and Private Working Folder](#)

[Preferred web scheme/skin](#)

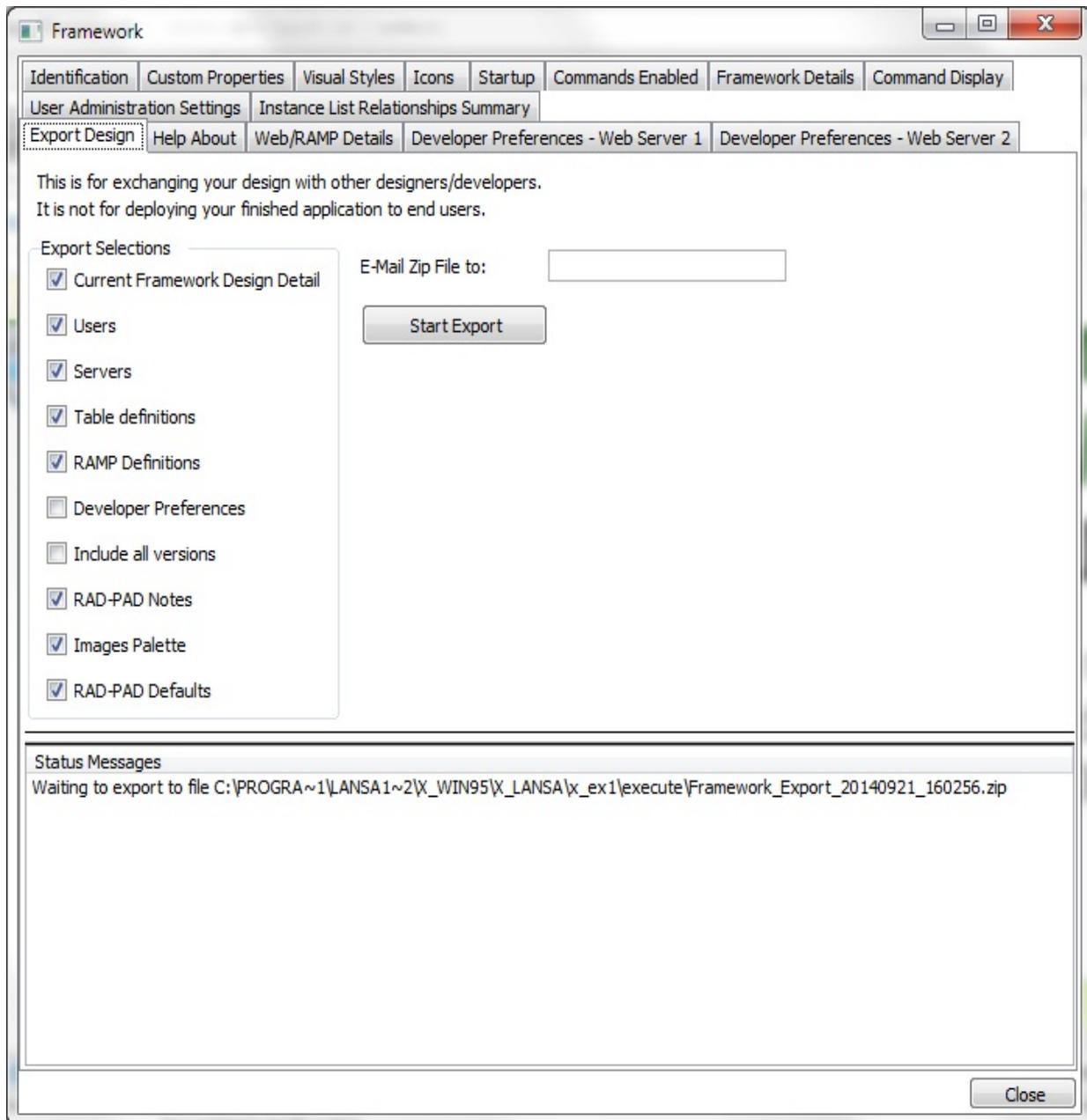
[Private Working Folder](#)

[Script for Uploading to your LANSA for the Web Folders](#)

[Web Server Caption](#)

Export Design

Use this tab to export your Framework:



[Email Zip File To](#)

[Export Developer Preferences](#)

[Export Framework Design](#)

[Export Images Palette](#)

[Export Include All Versions](#)

[Export RAD-PAD Defaults](#)

[Export RAD-PAD Notes](#)

[Export RAMP Definitions](#)

[Export Servers](#)

[Export Tables](#)

[Export Users](#)

Filters

Use these tabs to define your filters:

[Filter Snap-in Settings](#) tab

[Icons](#) tab

[Identification](#) tab

Filter Snap-in Settings

Use this tab sheet to define your filter:

The screenshot shows the 'Business Object Properties - Organizations' dialog box with the 'Filter Snap-in Settings' tab selected. The dialog has a left sidebar with 'Hidden Filter for Organizations' and a main area with several sections:

- Stay Active:** A dropdown menu set to 'Yes'.
- Filter Handler:**
 - VLf-WIN - Windows:** Includes a radio button for 'Component Identifier' (selected) with the value 'DF_FILT8', and a radio button for 'Mock Up - RAD-PAD' with a long alphanumeric string.
 - VLf-WEB / VLf.NET - Web Browser:** Includes a radio button for 'WEBEVENT/Hidden Function (Id)' with the value 'DWFILT8' and 'Process: DW_PROC'. Below it are dropdowns for 'Location for Buttons' (set to 'Right') and 'Type of Layout Style to be Used' (set to 'Header Panel and Browse List Panel'). It also has a radio button for 'WAM Identifier' (selected) with the value 'DM_FILT8', and radio buttons for 'AJAX' (with 'HTML Page' and 'Module' fields) and 'Mock Up - RAD-PAD' (with a long alphanumeric string).
 - .NET Component:** Includes fields for 'Class Name' and 'Assembly'.

At the bottom left are 'New' and 'Delete' buttons, and at the bottom right is a 'Close' button.

*WEBEVENT Function and/or WAM Component or AJAX Routine

AJAX Page (HTML File)

Associated AJAX Function

Component Identifier

Location for Buttons

[Mock Up RAD-PAD](#)

[.NET Component Class Name and Assembly](#)

[Stay Active](#)

[Type of Layout Style to be Used](#)

See also:

[Filter](#)

[Mock Up Filters and Command Handlers](#)

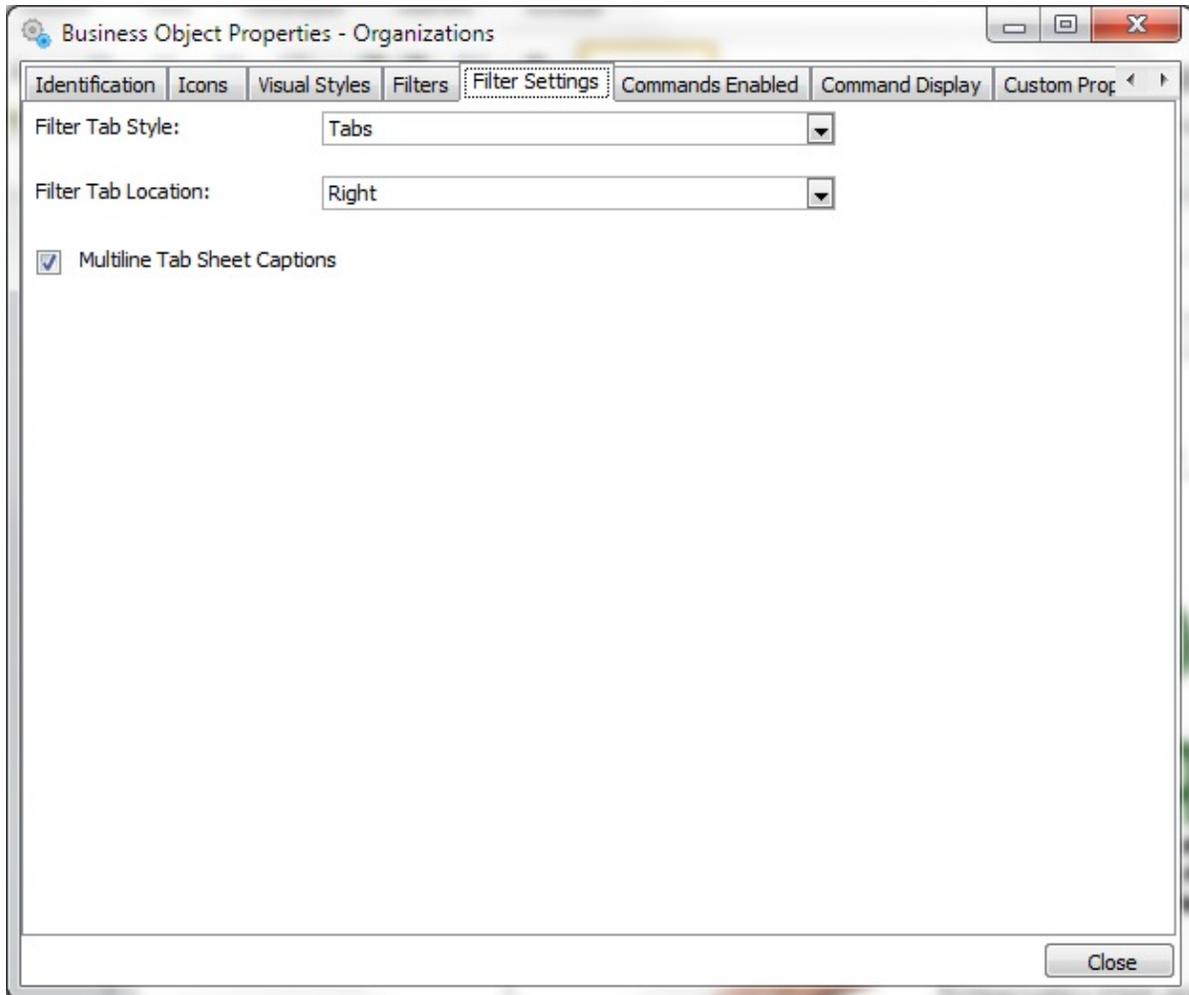
[Images Palette](#)

[Prototype Your Filters](#)

[Create Your Own Filters](#)

Filter Settings

Use this tab sheet to define how your filter is displayed:



[Filter Tab Style](#)

[Filter Tab Location](#)

[Multiline Tab Sheet Captions](#)

Framework Details

Use this tab to set options for the entire Framework:

Developer Preferences - Web Server 2 | User Administration Settings | Instance List Relationships Summary

Identification | Custom Properties | Visual Styles | Icons | Startup | Commands Enabled | **Framework Details** | Command Display | Export Design | Help About | Web/RAMP Details | Developer Preferences - Web Server 1

Framework Design and Run Time Properties

Enable Framework for Web Browser Applications Web Configuration Assistant

Enable Framework for WEBEVENT Functions

Enable Framework for WAMs (Web Application Modules)

Enable Framework for AJAX style applications

Compile Framework as Microsoft .NET Executable

Allow Panes to be Shrunk and Expanded

Enable the Position Menu Option

Encrypt XML files

Enable Development Status feature

Server Settings XML File:

Table definitions XML file

Nodes XML File

Developer Preferences XML File

Address for Error Notification:

Default Font when Printing Report Using Windows:

Automatic Save Time (in minutes)

Tool Bar Height

Tool Bar Style

Allow Search/Recently Used Limit

Search Field Width

Stay Active Default for Commands and Filters

Trim Working Set

Application Navigation Pane Views

- View as Two Lists (Side by Side)
- View as Two Lists (Over and Under)
- View as a Single Tree
- Launch from Status Bar
- View as Drop Down on Tool Bar
- Allow Users to Switch Views

Keep SID file versions (RAMP-NL only)

Keep XML File Versions

Keep Versions in Sub-Folders

Icon and Bitmap Encoder: (Id)

MTXT string loader (Id)

User Imbedded Interface Point: (Id)

XML Encoding

VLF.NET Options

Referenced .NET assemblies (for snap-in components)

.NET Target Platform

VLF.NET Screen Layout Persistence Level

Allow dynamic overriding of default application texts

Development Status Captions

Closed
Locked
Changed
Under Design
Under Implementation
Coding Complete
Under Test
Test Complete
Completed
Spare

Address for Error Notification
Allow Dynamic Overriding of Default Application Texts
Allow Panes to be Shrunk and Expanded
Allow Search/Recently Used Limit
Allow Users to Switch Views
Automatic Save Time in Minutes
Compile Framework as Microsoft .NET 2.0 Executable
Default Font when Printing a Report Using Windows
Developer Preferences XML File
Development Status Captions
Enable Development Status Feature
Enable Framework for AJAX style applications
Enable Framework for WAMS
Enable Framework for Web browser Applications
Enable Framework for WEBEVENT Functions
Enable the Position Menu Option
Encrypt XML Files
Icon and Bitmap Enroller
Keep Versions in Subfolders
Keep XML File Versions
Launch from Status Bar
MTXT String Loader
.NET Target Platform
Nodes XML File
RAD-PAD File Format
Referenced .NET Assemblies
Search Field Width
Server Settings XML File
Stay Active Default for Command Handlers and Filters.
Table Definitions XML File
Tool Bar Height
Tool Bar Style

Trim Working Set

User Imbedded Interface Point

View as A Single Tree

View as Drop Down on Toolbar

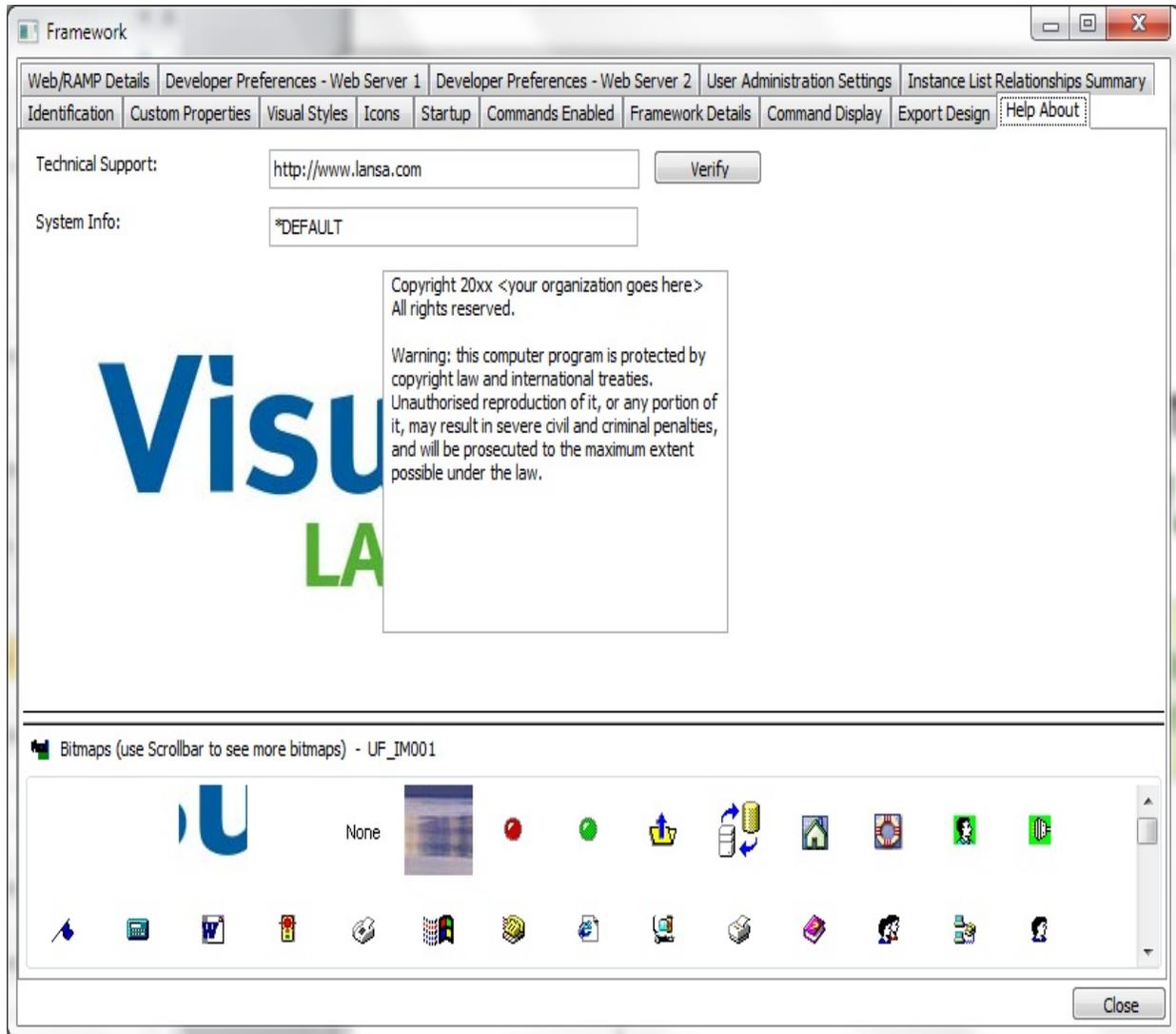
View as Two Lists Over and Under

View as Two Lists Side By Side

VLF.NET Screen Layout Persistence Level

Help About

This tab sheet contains properties:



Bitmap

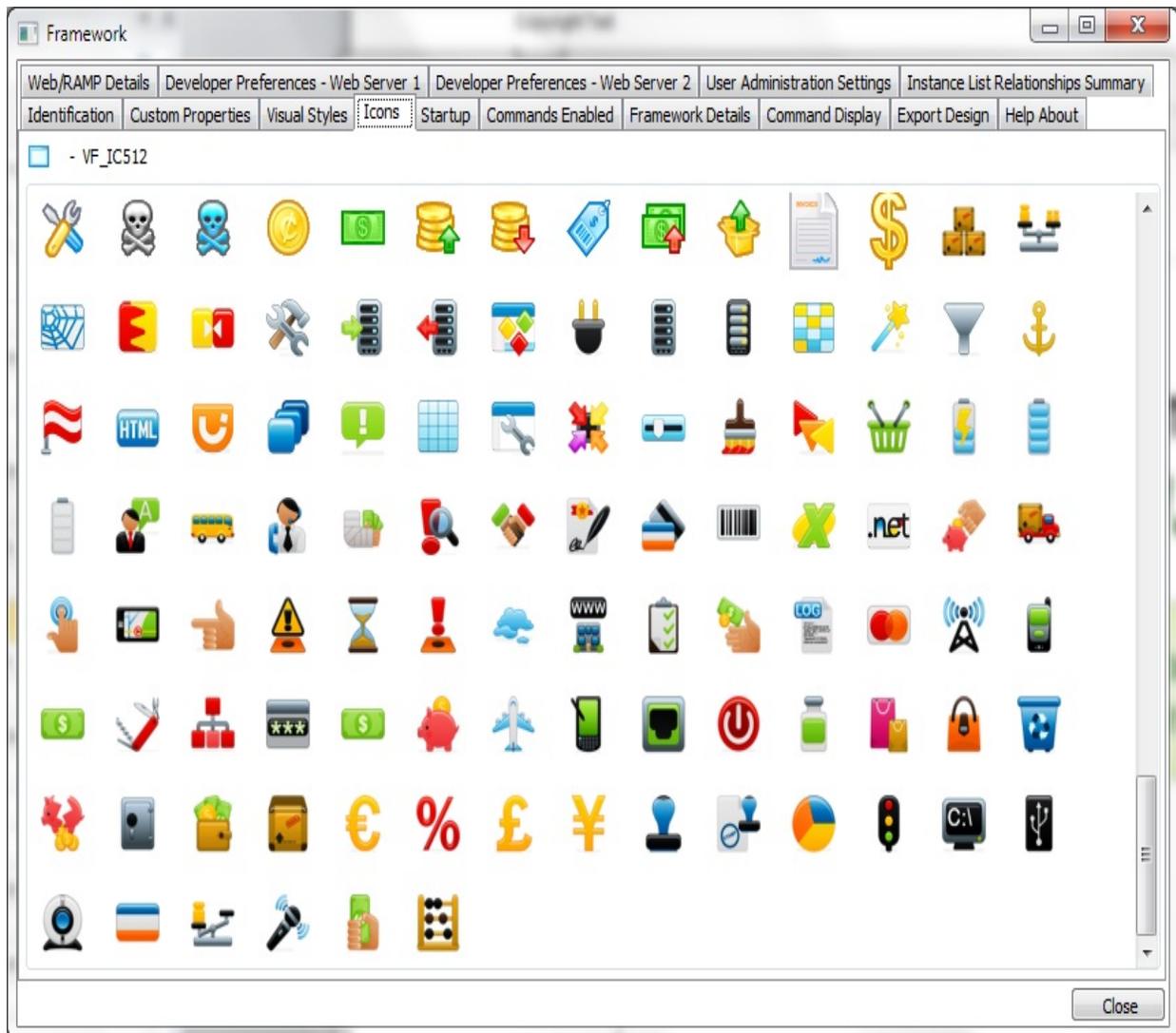
Copyright Text

Technical Support

System Info

Icons

Use this tab sheet to set icons:



Icon

Identification

Use this tab to identify your object in the Framework:

The screenshot shows a window titled "Framework" with a tabbed interface. The "Identification" tab is selected. The window contains the following fields and options:

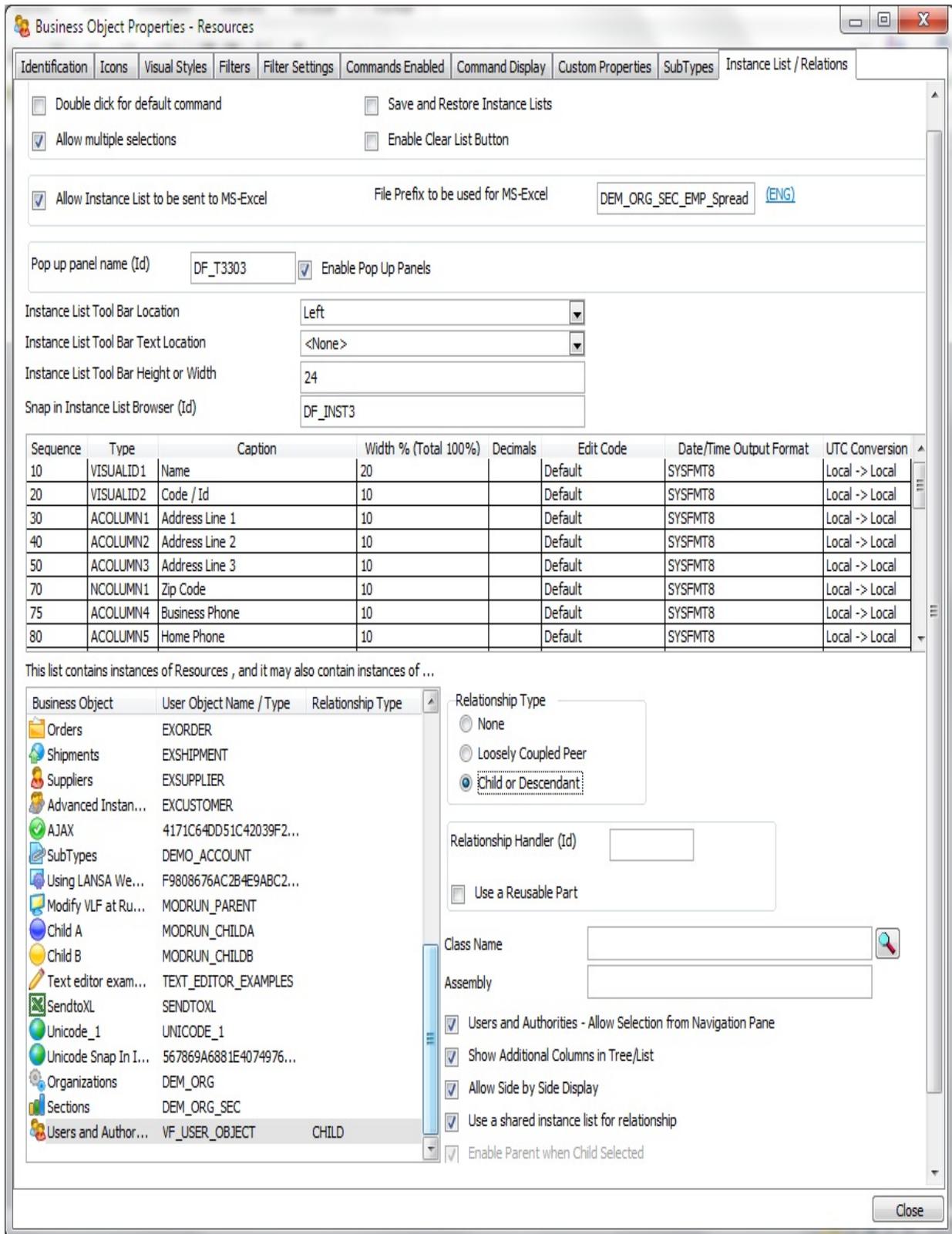
- Web/RAMP Details** | **Developer Preferences - Web Server 1** | **Developer Preferences - Web Server 2** | **User Administration Settings** | **Instance List Relationships Summary**
- Identification** | **Custom Properties** | **Visual Styles** | **Icons** | **Startup** | **Commands Enabled** | **Framework Details** | **Command Display** | **Export Design** | **Help About**
- Caption:** Your Framework (with a link to [\(ENG\)](#))
- Hint:** (empty text box)
- Sequence:** 1
- Internal Identifier:** VF_SHIPPED
- Unique Identifier:** 1
- User Object Name / Type:** SHIPPED_FRAMEWORK
- Show the 'Windows' Menu in this Framework
- Show Current Business Object in Window Title
- Allow this Object to be Opened in a New Window:** Manually
- Number of Additional Windows a User can have Open Concurrently:** 20
- Multiple Window Control Bar Location:** Above Title Bar
- Your Framework Version Number:** 1 . 0 . 0 . 132
 - Automatically Increment when Saving
 - Show in Help About Text
- Last Changed:** 20140929-152210-VLXPGLIB
- Alter Development Status** (dropdown menu)
- Close** (button)

Note that not all Identification tab properties are available for all object types.

Allow Selection from Navigation Pane
Allow on Web
Allow in Windows
Allow this Object to be Opened in a New Window
Alter Development Status
Caption
Caption (Singular)
Caption with Accelerator
Contains Favorites
Hint
Internal Identifier
Last Changed
Number of Additional Windows a User can have Open Concurrently
Multiple Window Control Bar Location
Restricted Access
Sequence
Shortcut
Show the 'Windows' Menu in this Framework
Show Current Business Object in Window Title
Unique Identifier
User Object Name/Type
Your Framework Version Number

Instance List/Relations

This tab sheet contains properties:



Allow Instance List to be sent to MS-Excel

Allow Multiple Selections

Allow Selection from Navigation Pane

Allow Side by Side Display

Business Object List

Columns for Instance Lists (Sequence, Type, Caption, Width, Decimals, Edit Code, Date/Time Output Format and UTC conversion.)

Double Click for Default Command

Enable Child when Parent Selected

Enable Clear List Button

Enable Parent when Child Selected

Enable Peers when Selected

Enable Popup Panels

File Prefix to be used for MS-Excel (Business object properties, Instance List tab)

Instance List Tool Bar Height or Width

Instance List Tool Bar Location

Instance List Tool Bar Text Location

Popup Panel Name

Relationship Handler

Relationship Type

Use a Reusable Part

Save and Restore Instance Lists

Show Additional Columns

Snap in Instance List Browser ID

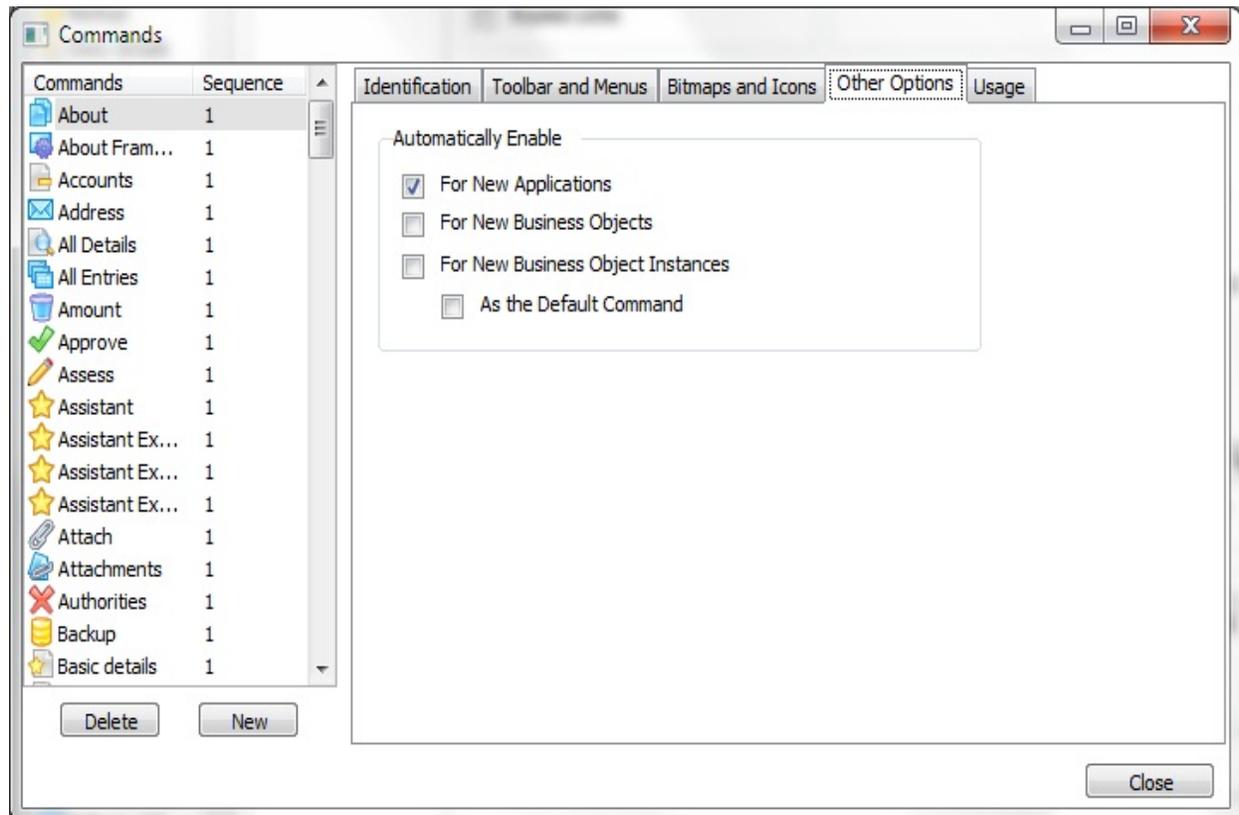
Use Shared Instance List for Relationships

Instance List Relationships Summary

Name	User Object Name / Type
<input type="checkbox"/> The essential business object	9A475843B9D5452D9A4926BDE8E5C927
<input type="checkbox"/> The CRUD business object	34C9245FB4C74621A893CCF65D11F2B0
<input type="checkbox"/> Selected, Current or All Entries	BD8D1D64689F468E8FA84DA2DA1A87FA
<input type="checkbox"/> Switching	A1DC6AABA4534FFFB30B6914B5B3D802
<input type="checkbox"/> Using the whole page	DF48013C5EF84E22991AEF8296367B72
<input type="checkbox"/> Showing a web page	A5DFC3E665EB4A29B729A1FE3DA1501D
<input type="checkbox"/> Handling Delete	380F2EC3946D44C4AB3086A68B0643BD
<input type="checkbox"/> Snap in Instance Lists	021168D7960C49CBA4A331132E1E1CE0
<input type="checkbox"/> Auto filling the instance list	BFD0C389C1C548579F55041AFF88F827
<input type="checkbox"/> Passing information	885BD9B508E0457190056ADA52765DA4
<input type="checkbox"/> Remembering values	A4DB7D8154424DA0BB13DA3BC50DB50C
<input type="checkbox"/> Debug using Trace	49765C6B7F37419ABA904BA8506CEA07
<input type="checkbox"/> End User Help (F1)	24C5B4B2FA2B485FB561B5741439C0D0
<input type="checkbox"/> Event Handling and Prompting	17A72AEB73AE4040A7BBA07EA2CD7827
<input type="checkbox"/> Using Code Tables	A4F9BCF013A348939CE2ADB86B0F607
<input type="checkbox"/> View iSeries Spool Files	F389D053D9F44DA2BB8791DCEB58D925
<input type="checkbox"/> Mini Filter (Simple)	C49756D18F504E54929CACD806669E9F
<input type="checkbox"/> Mini Filter (Multiple)	10D11D33C0B247169E2E18535E36DD3E
<input type="checkbox"/> Mini Filter (Multiple 2)	B694791D21C843F1AC39EEAE2BA0D1C9
<input type="checkbox"/> Advanced Instance Lists (1)	EXPRODUCT
<input type="checkbox"/> Orders	EXORDER
<input type="checkbox"/> Shipments	EXSHIPMENT
<input type="checkbox"/> Suppliers	EXSUPPLIER
<input type="checkbox"/> Advanced Instance Lists (2)	EXCUSTOMER
<input type="checkbox"/> AJAX	4171C64DD51C42039F2929E574A821A1
<input type="checkbox"/> SubTypes	DEMO_ACCOUNT
<input type="checkbox"/> Using LANSAs Weblinks	F9808676AC2B4E9ABC2F5463E8963A91
<input type="checkbox"/> Modify VLF at Runtime	MODRUN_PARENT
<input type="checkbox"/> Child A	MODRUN_CHILDA
<input type="checkbox"/> Child B	MODRUN_CHILDB
<input type="checkbox"/> Text editor examples	TEXT_EDITOR_EXAMPLES
<input type="checkbox"/> SendtoXL	SENDTOXL
<input type="checkbox"/> Unicode_1	UNICODE_1
<input type="checkbox"/> Unicode Snap In Instance List	567869A6881E4074976D8820126E126F
<input type="checkbox"/> Organizations	DEM_ORG
<input type="checkbox"/> Sections	DEM_ORG_SEC
<input type="checkbox"/> Resources	DEM_ORG_SEC_EMP
<input type="checkbox"/> Users and Authorities	VF_USER_OBJECT
<input type="checkbox"/> New Business Object	D3E1DE0BDF3540F4BE1D5086BF039BFA
<input type="checkbox"/> New Business Object	EA284D78A7CB4471B1ADBAC1ACEEF403
<input type="checkbox"/> New Business Object	40AA7E9B8CF34917A5A314CE312A5EA2

Close

Other Options



As the Default Command

Automatically Enable for New Applications

Automatically Enable for New Business Object Instances

Automatically Enable for New Business Objects

Server Details

The Server Details tab contains these properties:

The screenshot shows a window titled 'Servers' with a table on the left and a 'Server Details' tab on the right. The table has two columns: 'Sequence' and 'Server'. The first row contains '1' and 'MY IBM i'. The 'Server Details' tab is divided into several sections:

- Server Type:** LANSA for System i (dropdown)
- Server Name:** VLXPGMLIB (text box)
- Partition:** EX1 (text box)
- Client-Server Translation Table:** *JOB (text box)
- Server-Client Translation Table:** *JOB (text box)
- Selection Block Size:** 500 (text box)
- Selection Limit:** 10000 (text box)
- Execution Priority:** 20 (text box)
- Server Overrides:** (empty text box)
- Server IIP func to validate sign on (Id):** UFU0005 (text box)
- DBCS Capable:**
- Commitment Control:**
- Divert Locks:**
- Partition is enabled for RDM:**
- Use Windows Credentials:**
- Upper and Lower Case Pas:**
- Show on Connect Dialog:**

Below these are two more sections:

- IBM i Host Server Mapper:**
 - Name / IP address:** VLXPGMLIB (text box)
 - Port:** 5057 (text box)
- Recover Connection:**
 - Attempt automatic session recovery
 - Time interval between checks of connection status (seconds):** 30 (text box)
 - Check connection before executing commands
 - Check connection before selecting applications and business objects
 - Action to take when session cannot be recovered:** Notify and allow retry (dropdown)
 - Check connection using function (Id):** UFU0004 (text box)

At the bottom left are 'New' and 'Delete' buttons. At the bottom right are 'LANSA Comms. Admin' and 'Close' buttons.

The properties shown on the Server Details tab depend on the type of server chosen.

[Attempt Automatic session recovery](#)

[Action to take when session cannot be recovered](#)

Check connection before executing commands
Check connection before selecting applications and business objects
Check Connection using function
Client Server Translation Table
Codebase
Commitment Control
Database Password
Database Type
Database User
DBCS Capable
Divert Locks
Engine
Execution Priority
IBM i Host Server Mapper Name / IP address
IBM i Host Server Mapper Port
IP Address and Port Number
Load Path
Partition
Partition is Enabled for RDMLX
Private Definition/aXes Project Folder
RAMP Tools Mode Load Path
Save as Deployment Server
Selection Block Size
Selection Limit
Server Client Translation Table
Server IIP function to validate sign on
Server Name
Server Overrides
Server Type
Show on Connect Dialog
Time interval between checks of connection status
Update File

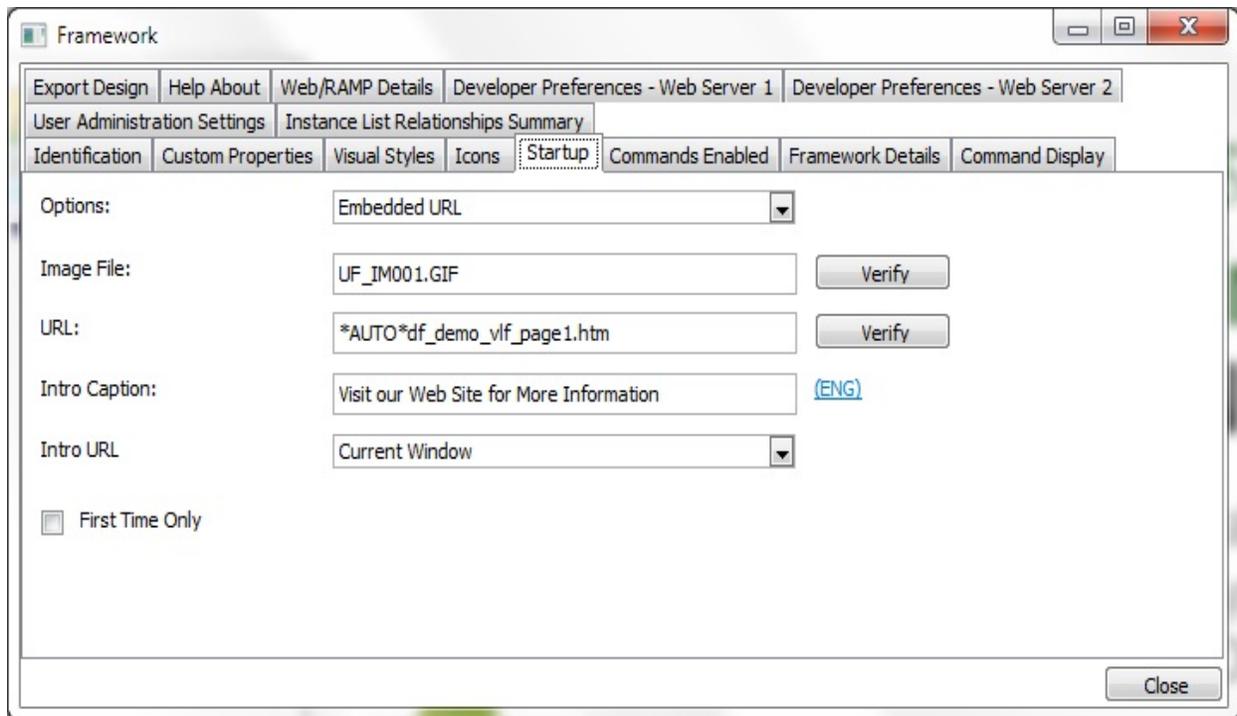
Upper and Lower Case Password

Use HTTPS

Use Windows Credentials

Startup

Use this tab sheet to define how your Framework starts up:



[First Time Only](#)

[Image File](#)

[Intro Caption](#)

[Intro URL](#)

[Options](#)

[URL](#)

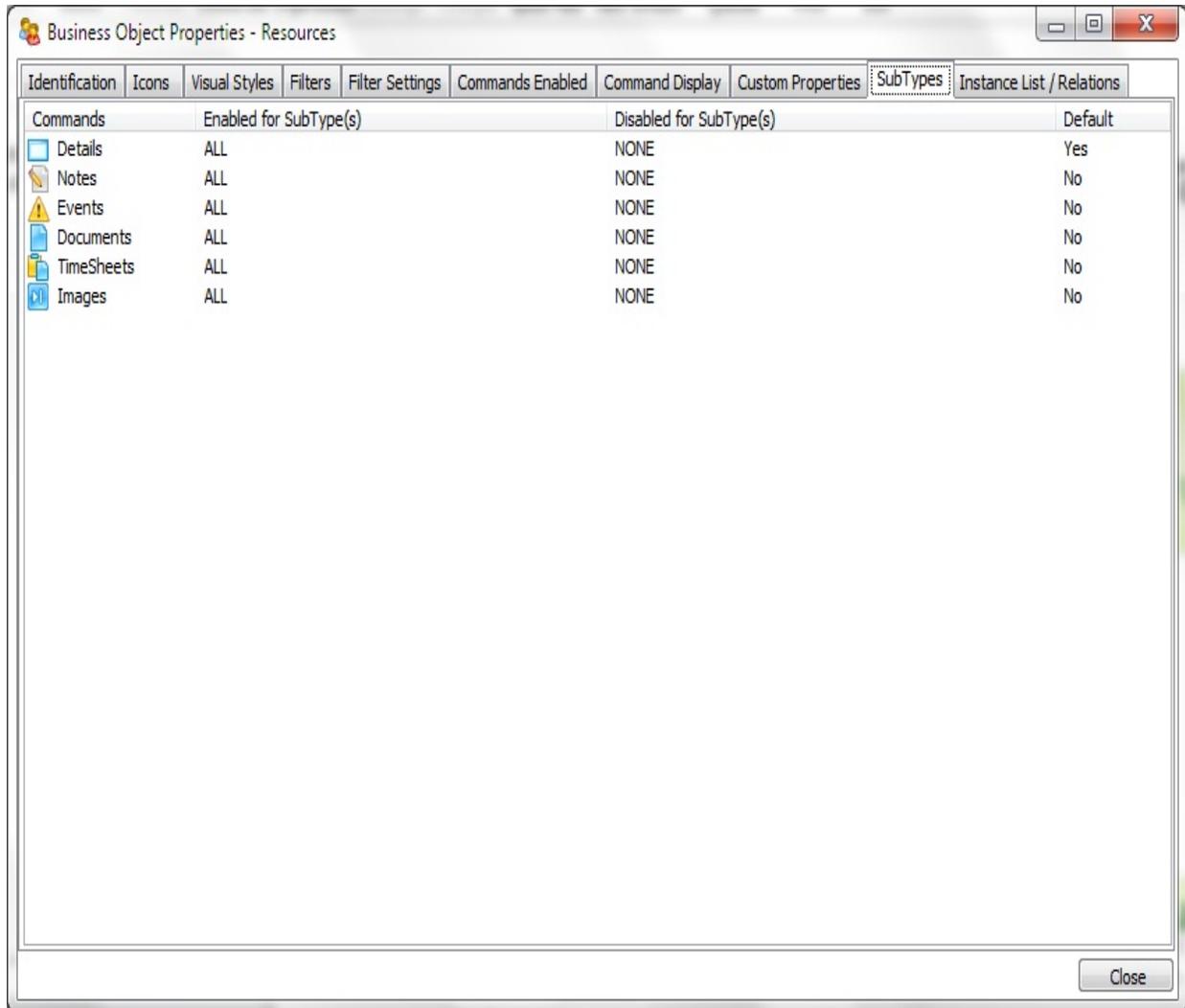
See also:

[How do I change introduction logos?](#)

[How do I change the logo shown when the Framework starts executing?](#)

Subtypes

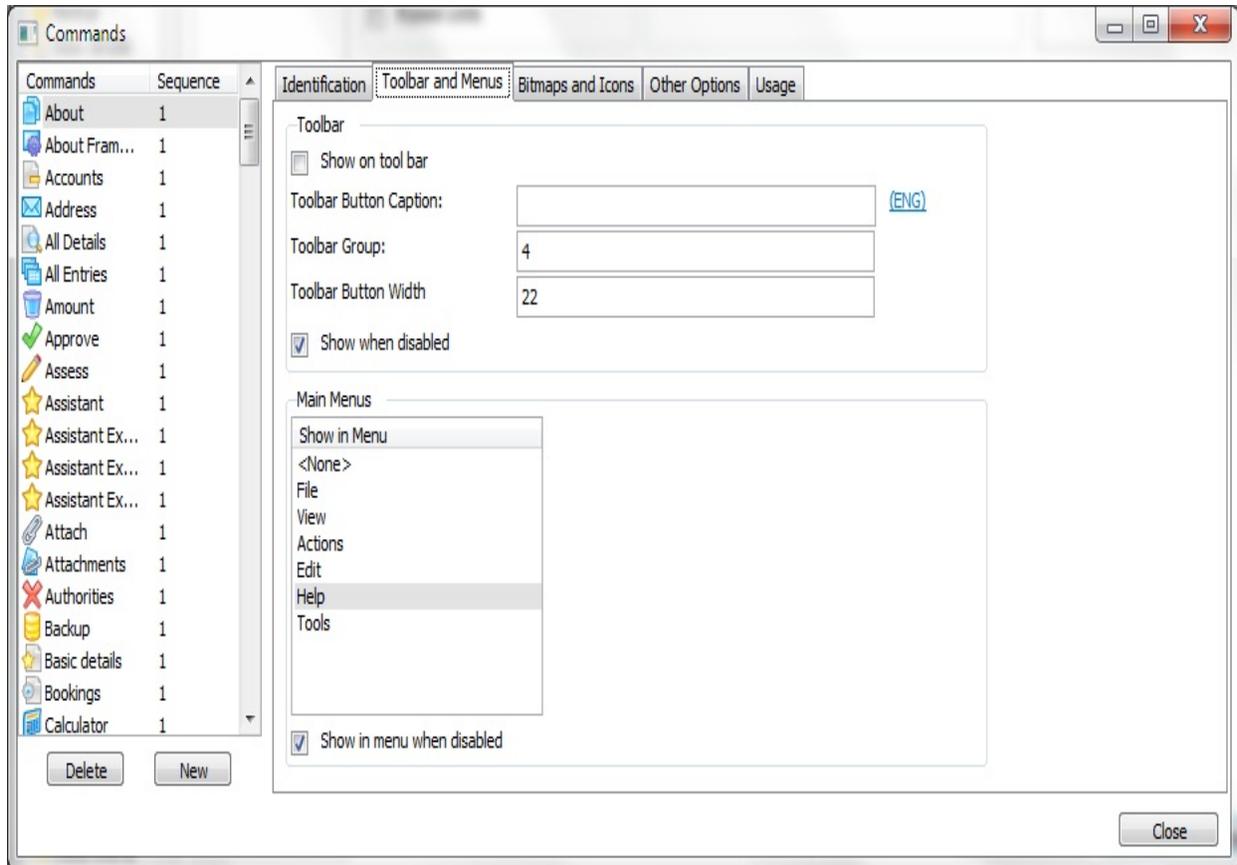
Use this tab sheet to specify business object subtypes:



See [SubTypes](#).

Toolbar and Menu

This tab sheet contains properties:



Show in Menu

Show When Disabled

Show on Toolbar

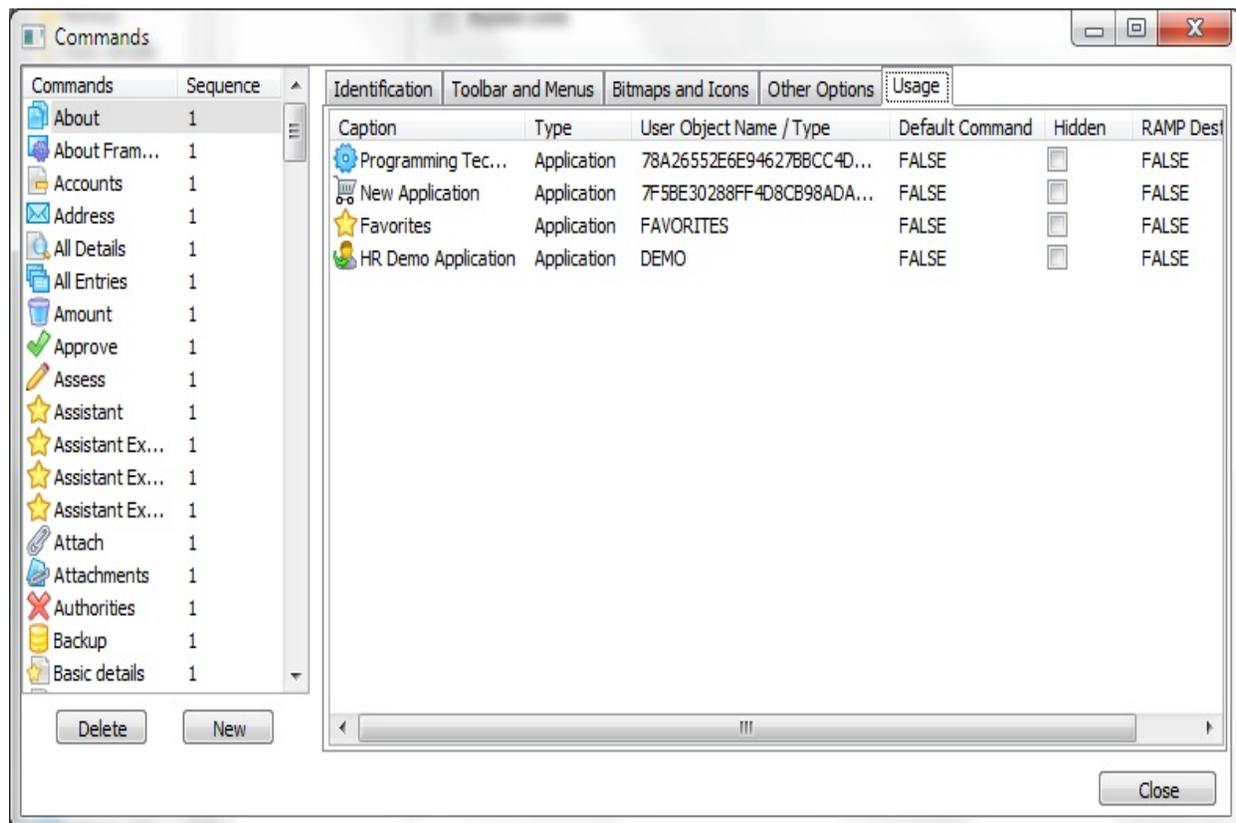
Show When Disabled

Toolbar Button Caption

Toolbar Button Width

Toolbar Group

Usage



The tab lists the business objects, applications and Frameworks in which a selected command is used. It also shows whether the command is a default command, whether it is hidden and whether it is attached to a RAMP destination screen.

User Administration Settings

Use this tab to specify user administration settings for the Framework.

The screenshot shows a dialog box titled "Framework" with a tabbed interface. The "User Administration Settings" tab is selected. The dialog is organized into several sections:

- Authority Settings:** Includes a checked checkbox for "Use Framework Users and Authority", an unchecked checkbox for "Store Users in DBMS Tables VFPPF06/7", a text field for "Store Users in XML File Named" containing "Vf_Sy001_Users.xml", and two empty text fields for "Import Users Imbedded Interface Point (Id)" and "Report on Users Imbedded Interface Point (Id)".
- Sign on Settings:** Features a dropdown menu for "End Users must Sign on to this Framework" set to "Never (no signon is required for any application)". It has two radio button options: "Users Sign on Locally to Use the Framework" (selected) and "Users Sign on to a Remote Server to Use the Framework". Under the local option, there is a checked checkbox for "User Can Change Own Password" and a "Maximum Signon Attempts Allowed" field set to "6". A sub-section "If Maximum Allowed Sign on Attempts Exceeded" has two radio buttons: "Advise User with a Message" (selected) and "Framework Fatal Error".
- Sign on Form Settings:** Contains text fields for "Launch Button Caption" (set to "Start"), "Launch URL (Windows)", and "Launch URL (Web / .Net)". Each URL field has a "Verify" button. A sub-section "VLF-WIN - Windows" includes a text field for "Image / Web Page to Display" (set to "*AUTO*df_demo_vlf_page2.htm"), a dropdown for "Alignment of Image on Form" (set to "Centered"), and text fields for "Height" (145) and "Width" (350).
- IBM i User Profile Management:** Includes a checked checkbox for "Allow IBM i password change", a text field for "Change Password IIP" (set to "UFU0006"), and an "Extended validation" section with a checked checkbox for "Check Password Expiry" and a "Warn before (days)" field set to "10".
- Timeout Settings:** Contains two empty text fields for "Inactivity Log on timeout (minutes)" and "Inactivity Log off timeout (minutes)".

A "Close" button is located at the bottom right of the dialog.

Advise User With a Message

Alignment of Image on Form
Allow IBM i password change
Change Password IIP
Check Password Expiry
End-users Must Sign on to this Framework
Framework Fatal Error
Height
Image / Web Page to Display on Form
Import Users Imbedded Interface Point
Launch Button Caption
Launch URL (Windows)
Launch URL (Web / .Net)
Log off Inactivity Timeout
Log on Inactivity Timeout
Maximum Signon Attempts Allowed
Name of User Set to be Used
Report on Users - Imbedded Interface Point (Id)
Store Users in XML File and Store users in DBMS Tables VFPPF06/07
Use Framework Users and Authority
User Can Change Own Password
Users May Work Offline if the Remote Server Is Not Available
Users Sign on Locally to Use the Framework
Users Sign on to a Remote Server to Use the Framework
Warn before (days)
Width

See also:

Users

User Details

This tab sheet contains properties:

The screenshot shows a window titled 'Users' with a list of users on the left and a 'User Details' tab selected on the right. The list of users is as follows:

Group?	User Profile	Caption
	_COPYUSR	New User
	VLXPGMLIB	New User

The 'User Details' tab for the selected user '_COPYUSR' contains the following fields and options:

- User Profile:
- New Password:
- Caption:
- Temporary Directory on PC:
- Email Address:
- Inactivity Log on timeout (minutes):
- Inactivity Log off timeout (minutes):
- Administrative User
- User is Disabled
- Groups this user belongs to:
- Only use when the User details for RAMP sessions need to be different
- RAMP User:
- RAMP Password (optional, rarely used):

At the bottom of the window, there are several buttons: 'New User', 'New Group', 'Delete', 'Copy', 'Save', 'User Authorities Report File', 'Export Users', 'Import Users from XML', and 'Close'.

Administrative User

Caption

Email Address

Groups this user belongs to

Log off Inactivity Timeout

Log on Inactivity Timeout

New Password

RAMP Password

RAMP User

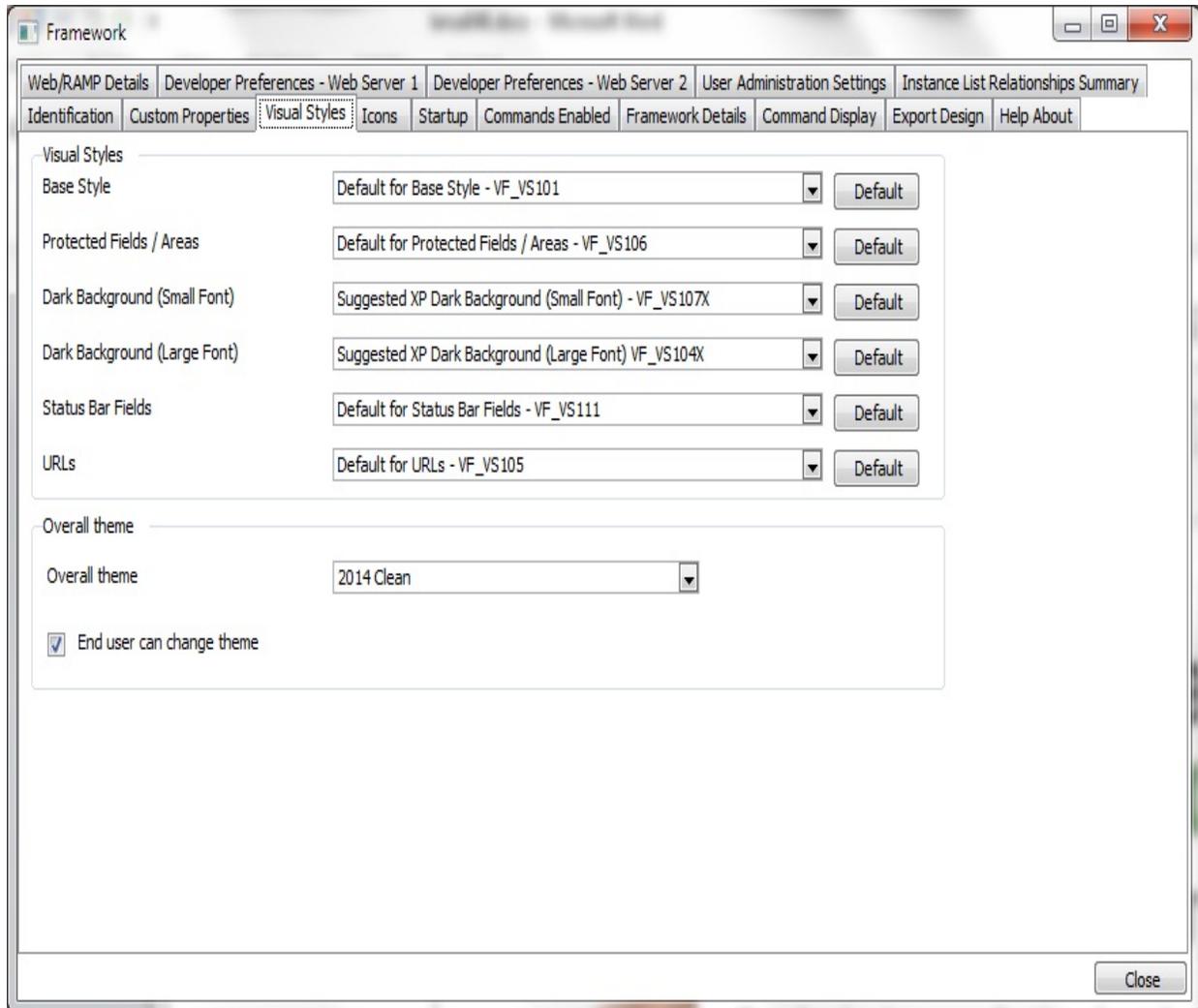
Temporary Directory on PC

User Is Disabled

User Profile

Visual Styles

Use this tab to define how your Framework is presented:



Note that not all Visual Styles tab properties are available for all object types.

[Visual Style Base Style](#)

[Visual Style Protected Fields and Areas](#)

[Visual Style Dark Background Small Font](#)

[Visual Style Dark Background Large Font](#)

[Visual Style Status Bar Fields](#)

[Visual Style URLs](#)

[Overall Theme](#)

End User can change theme

Before using visual styles, investigate the use of a **Overall Theme**. You may be able to get the appearance you want with fewer changes.

Visualization

Use this tab to describe how the table data will be presented if it is shown in a combo box or radio buttons:

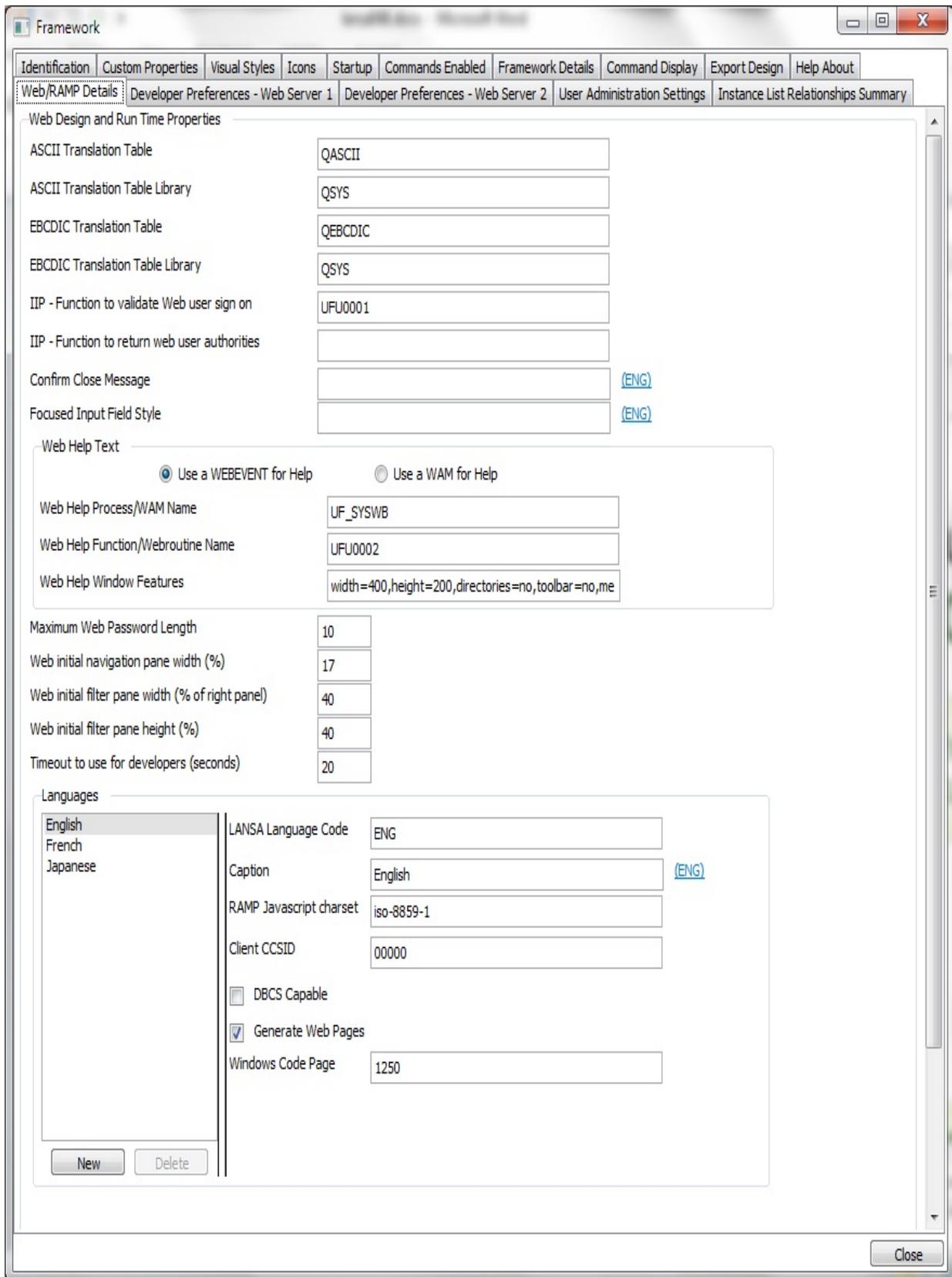
The screenshot shows a 'Table Design' window with the 'Visualization' tab selected. The table name is 'Departments'. The left pane lists the fields: Australian States, Sex, Titles, Country (Windows only), Currency (Windows only), and USA States. The right pane has a sub-header 'When visualized as a combo box or as radio buttons' and two dropdown menus: 'Displayed field' set to 'Description' and 'Sequence using' set to 'Code'. At the bottom are 'New', 'Delete', and 'Close' buttons.

Field	Displayed field	Sequence using
Australian States	Description	Code
Sex	Description	Code
Titles	Description	Code
Country (Windows only)	Description	Code
Currency (Windows only)	Description	Code
USA States	Description	Code

Displayed Field
Sequence Using

Web/RAMP Details

Use this tab to set design and run time properties for web applications:



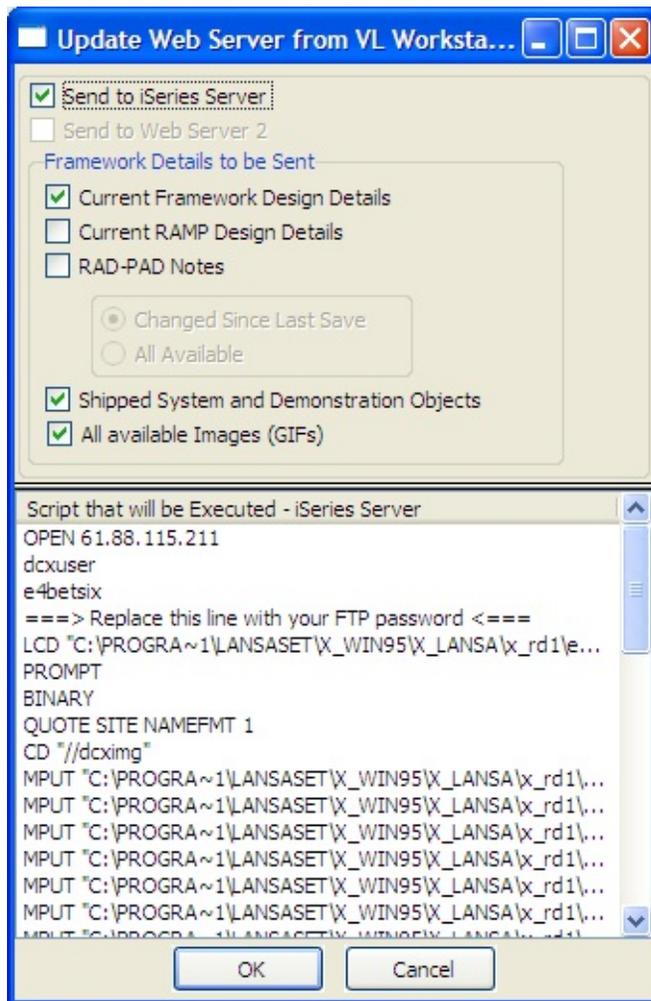
ASCII Translation Table Name and Library

Caption
Client CCSID
Close Confirmation Message
DBCS Capable
EBCDIC Translation Table Name and Library
Focused Input Field Style
Generate Web Pages
IIP – Function to return web user authorities
IIP - User Signon Function Name
Languages
LANSA Language Code
Maximum Web Password Length
Meta Tag
RAMP Javascript Charset
Timeout to use for developers
Use a Webevent/WAM for Help
Web Help Function/Webroutine Name
Web Help Process/WAM Name
Web Help Window Features
Web Initial Filter Pane height (%)
Web Initial Filter Pane width (% of right panel)
Web Initial Navigation Pane width (%)
Windows Code Page

Dialogs

Update Server from Visual LANSA Workstation

If your Framework is enabled for Web browser applications, this form is presented whenever you save changes you have made to your Framework design.



This form will send your Framework details to your selected LANSA for the Web server(s) so that they can be executed as part of a Web browser application.

The web server(s) to which the details will be sent to are indicated at the top of the panel. You can change the default server selection(s) by checking or unchecking the server name.

Normally the Framework will automatically check the correct options and you should just click the OK button.

If you have just enabled your Framework for browser applications or just upgraded your Framework version, you should always select "Shipped System and Demonstration Objects" and "All available Images (GIFs)" the very first

time that you save your changes to a particular web server.

When you click the OK button the script will be executed to upload details to your LANSAs for the Web server(s).

While this is happening, a Windows command window may be displayed. The time taken will depend upon the options you have selected and the type and speed of the communications link you have with your LANSAs for the Web server. When the script has completed execution the window is displayed showing the script results.

You should scroll through the results checking for errors.

For more information about the upload script and your associated LANSAs for the Web server(s) refer to the [Developer Preferences – Web Server](#) tabs in your Framework properties by using menu options (Framework) -> (Properties).

The options available on this form are:

[Current Framework Design Details](#)

[RAD-PAD Notes](#)

[Shipped System and Demonstration Objects](#)

[All available Images \(GIFs\)](#)

Current Framework Design Details

Check this option to upload your Framework design and associated start up web pages to your LANSA for the Web server. Normally this option is always checked.

RAD-PAD Notes

Check this option to upload your RAD-PAD notes to your LANSAs for the Web server.

You can choose to upload all existing RAD-PAD notes or only the ones that you have changed since you last saved the Framework (within the current design session). Normally this option is checked automatically by the Framework and you would only ever manually check it to force an upload of all available RAD-PAD notes to correct a synchronization problem.

Shipped System and Demonstration Objects

Check this option to upload all Framework web components to your LANSA for the Web server.

This option is normally only ever checked immediately after enabling the Framework for browser applications or after performing a Framework version upgrade. Using this option may very significantly increase upload times.

All available Images (GIFs)

Check this option to upload all GIF image files from your current partition EXECUTE directory to your LANSA for the Web server. You would normally only select this option immediately after enabling the Framework for browser applications or after performing a Framework version upgrade. Using this option may very significantly increase upload times.

Execute Framework as a Web Application

Execute Framework as Web Application - Web Server 1

Choose Browser

Chrome

Firefox

Safari

Internet Explorer

Start

Inside Web Browser

In Own Window

Turn Tracing On

Touch Device

In Language

ENG - English

URL:

OK Cancel

If you are a designer and your Framework is enabled for web applications then this form is presented when you use the (Framework) then (Execute as Web Application ...) menu options.

It allows you to launch your Framework as web application.

Simply choose the options you want to use and click the OK button. The options available on this form are:

Choose Browser

Start

RAMP Application Testing

Turn Tracing On

In Language

Touch Device

URL

Choose Browser

See [Using VLF-WEB Applications with Safari, Firefox or Chrome](#).

Choose the browser in which you want to execute the Framework:

- Chrome
- Firefox
- Safari
- Internet Explorer

·

Start

Elect to start your application as VLF.WEB (Web Browser) or VLF.NET (.NET application). For VLF.WEB applications you may choose whether to start your application inside an Internet Explorer window or in its own window.

RAMP Application Testing

This group box appears if you are executing an application that may use RAMP.

The server name should specify a System i server that has been previously defined as the RAMP server. The user and password fields will be used to connect the System i server when required.

Note: Using this option adds the user details to the URL that will be opened in your web browser. This is a simple and convenient way to perform RAMP application testing. However, it would be unusual to start a RAMP application this way in an end-user application.

Turn Tracing On

Select this option to trace your application. A trace window will be displayed and it will show you what your application is doing on the client (within IE) and on the server. Refer to the "Debugging your Applications" section for more details about tracing.

Selecting this option is identical to adding ?Trace=Y to end of any Framework URL.

In Language

Choose the language in which the Framework is to be executed.

This option applies to VLF.WEB and VLF.NET applications.

This property is in the [Web/RAMP Details](#) tab.

URL

This is the URL that will be opened when you click the OK button. As you change the options on this form the URL will be changed to reflect your selections.

The URL is an input area so that:

- It easy for you to copy the URL (for example to email it someone else who you want to execute your Framework or to make it the command executed by an icon on your desktop).
- So that you can add special options to it. For example, your product vendor might ask you to add ?Trace=System to the end of the URL when attempting to diagnose a problem.

Touch Device

Adds TOUCH=Y to the URL.

Using TOUCH=Y enables framework functionality to make it more touch friendly. For an overview please refer to [Touch Device Considerations](#).

If the TOUCH parameter is not present in the URL the VLF will guess whether it is running on such device.

Properties

Add Fields from this Physical File

Specify the name of a physical file from which you want to add fields to this definition.

This is a [Program Coding Assistant](#) property.

Additional Columns for Building AColumn and NColumn Values

Specify any additional columns you wish to add to the instance list associated with this business object.

Any field name can be specified in this list (even ones not currently defined in the LANSAs data dictionary). The fields specified do not necessarily have to be columns in any physical file that you specify at the top of the form.

See [Visual Identifiers](#) and [Programmatic Identifiers](#) for more details of instance lists.

This is a [Program Coding Assistant](#) property.

Address for Error Notification

Specify here the email address to which a notification of fatal errors in your application are sent.

The Framework includes standardized [Application Error Handling](#). If you want the error handler to allow error details to be emailed by your end-users specify the email address here.

This option only applies to **Windows** Framework applications.

This property is in the [Framework Details](#) tab.

Administrative User

Use this option to indicate which users are administrative users.

An administrative user is a user who can maintain the lists of user profiles and servers. They use UF_ADMIN (or equivalent) as the entry point to the Framework.

This option is only checked by the Framework when the user must sign on to use the local database and the user initiated the Framework using UF_ADMIN (or equivalent). Users accessing the Framework through UF_DESGN (or equivalent) are not checked to be administrators.

Users accessing the Framework through UF_DESGN or UF_ADMIN (or equivalent) who use the Framework in connect to server mode or who do not sign in are not checked to see if they are administrators.

This property is in the [User Details](#) tab.

Advise User With a Message

This option only applies to **Windows** Framework applications.

When the user makes an unsuccessful signon attempt that exceeds the maximum allowed, a status bar message indicating that this user profile has been disabled is displayed. Further attempts to sign on as this user profile, with or without the correct password, will result in the same message.

To re-enable this user profile, the administrator must deselect the User Is Disabled checkbox for this user in the User Details tab.

This property is in the Framework [User Administration Settings](#) tab.

AJAX Page (HTML File)

An AJAX page is an HTML document. Typically it contains JavaScript to manage interactions between the application server and the user interface.

See [Framework-AJAX Applications](#).

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

Alignment of Image on Form

Select where the image will be displayed on the Sign On form. You can choose from Centred (default), Left or Right.

If a web page is used, this property is ignored.

This property is in the Framework [User Administration Settings](#) tab.

Allow Dynamic Overriding of Default Application Texts

Specifies whether default application texts (such as User and Password labels in the Sign On dialog) can be overridden dynamically in the production environment.



The default texts are overridden by modifying the VF_MULTI_YYY.js file (where YYY is the language code) and placing it on the webserver.

In the development environment, this file can be found on the LANSAs partition execute folder.

Note that this file needs to be present in the production environment.

This property is in the [Framework Details](#) tab.

Allow Generic Searching

Check this option if the filter is to support generic searching.

Technically this option simply means that SELECT commands generated will use the GENERIC(*YES) parameter.

Also see [Select the Keys of the Selected View to be used for Search Operations](#).

This is a [Program Coding Assistant](#) property.

Allow Float

Uncheck this option in situations where commands associated with this object are not allowed to float free in a separate form.

For example, if you are using a complex ActiveX control on your form and you find that it does not function correctly when floating free, use this option to stop the float operation.

This option only applies to **Windows** Framework applications.

This property is in the [Command Display](#) tab.

Allow IBM i password change

Check this option to allow IBM i password change. When this property is enabled, the Change IBM i Password button appears in the Logon panel in both Windows and Web.

This property is in the Framework [User Administration Settings](#) tab.

Allow in Windows

Uncheck this option if you do not want this application, business object or command to ever be usable in Windows applications.

Note that when you execute an application in Design mode, all command handlers and filters are visible, regardless of whether they have been enabled for Windows. To test how your application will appear to your end-user, execute it as an end-user.

This property is in the:

The [Commands Enabled](#) tab

The [Identification](#) tab

Allow Instance List to be sent to MS-Excel

This option allows end-users to send the contents of the instance list to a Comma Separated Variable (CSV) file. If the end-user has MS-Excel, then this will automatically display the data as a spreadsheet. The user can then save the spreadsheet where ever they want.

If this option is enabled, a new menu option becomes available when the end-user right mouse clicks on the instance list, called Send to MS-Excel. If the end-user takes this option, the current visible contents of the instance list are made into a CSV file.

The column and record ordering is the same as the user sees it on the instance list.

Visible child records are included but not indented.

If a side-by-side secondary instance list is visible and is showing data this data will also be included in the CSV file.

Default is checked for new business objects, and unchecked for pre-existing business objects.

Applies to **Windows**.

This property is in the Business Object [Instance List/Relations](#) tab.

Allow Multiple Selection

When the input method is an Alphanumeric or Numeric Fixed List use this option to indicate whether multiple items can be chosen from the fixed list.

This property is in the [Custom Properties](#) tab.

Allow Multiple Selections

This option indicates whether the user is allowed to concurrently select more than one item from an instance list.

The default is that they are allowed to. This option is only applicable to VLF.WIN applications using the shipped instance list browser.

If an instance list contains peer or child business objects, the Multiple Selections option for the primary/root business object is used.

This property is in the Business Object [Instance List/Relations](#) tab.

Allow on Web

Uncheck this option if you do not want this application, business object or command to ever be usable in Web browser applications.

Note that when you execute an application in Design mode, all command handlers and filters are visible, regardless of whether they have been enabled for Web. To test how your application will appear to your end-user, execute it as an end-user.

This property is in:

The [Commands Enabled](#) tab

The [Identification](#) tab

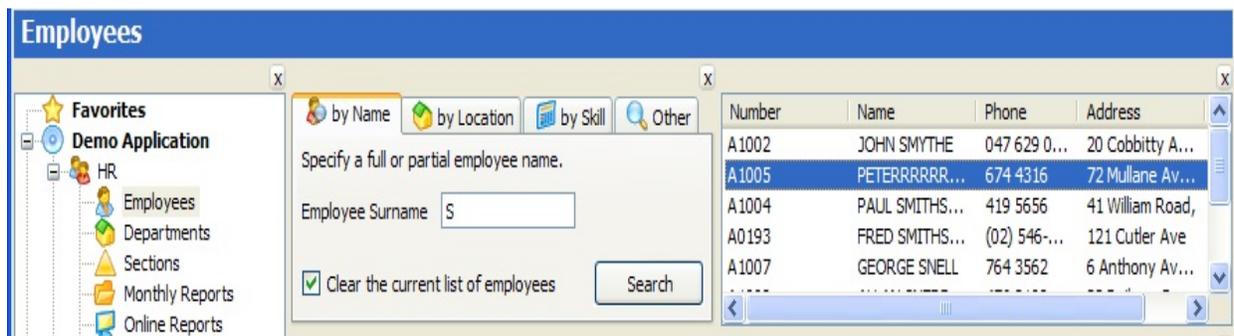
Allow Panes to be Shrunk and Expanded

Applies to **Windows** and **.NET**.

In Windows Framework applications:

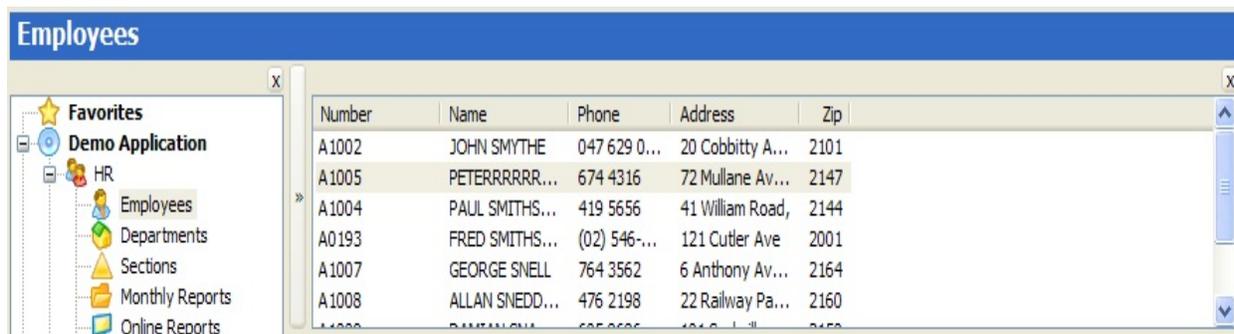
Use this option to indicate whether your application will allow the shrinking and expansion of the main Framework panes. When this option is enabled the major pane areas of the Framework display a small close button.

For example, here is a Framework application displaying the shrink/expand option in the navigation pane, the filter pane and the instance list pane:

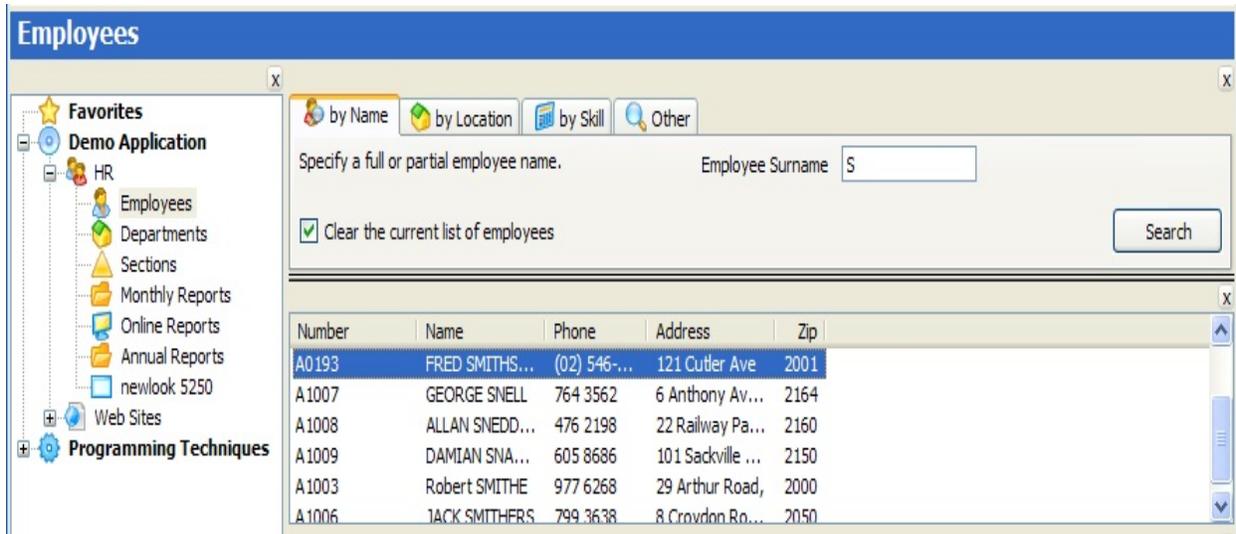


The user may click the button to shrink the pane when they do not need to use it.

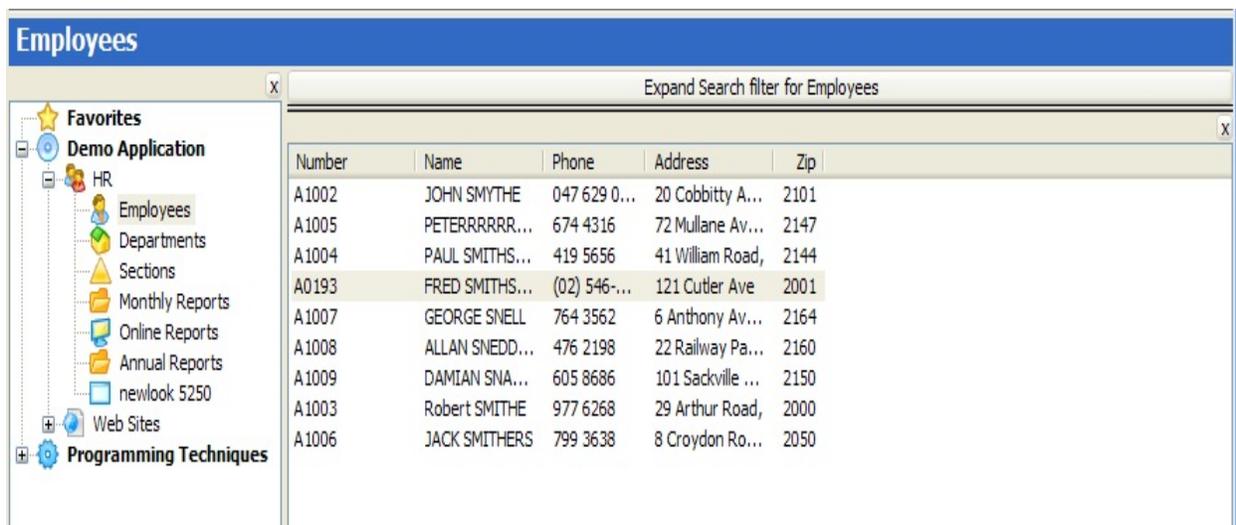
For example, here the user has shrunk the filter pane by clicking the button so that the instance list is easier to work with. The filter pane is reduced to a slim vertical bar displaying >>. To expand the pane again the user clicks the >> button:



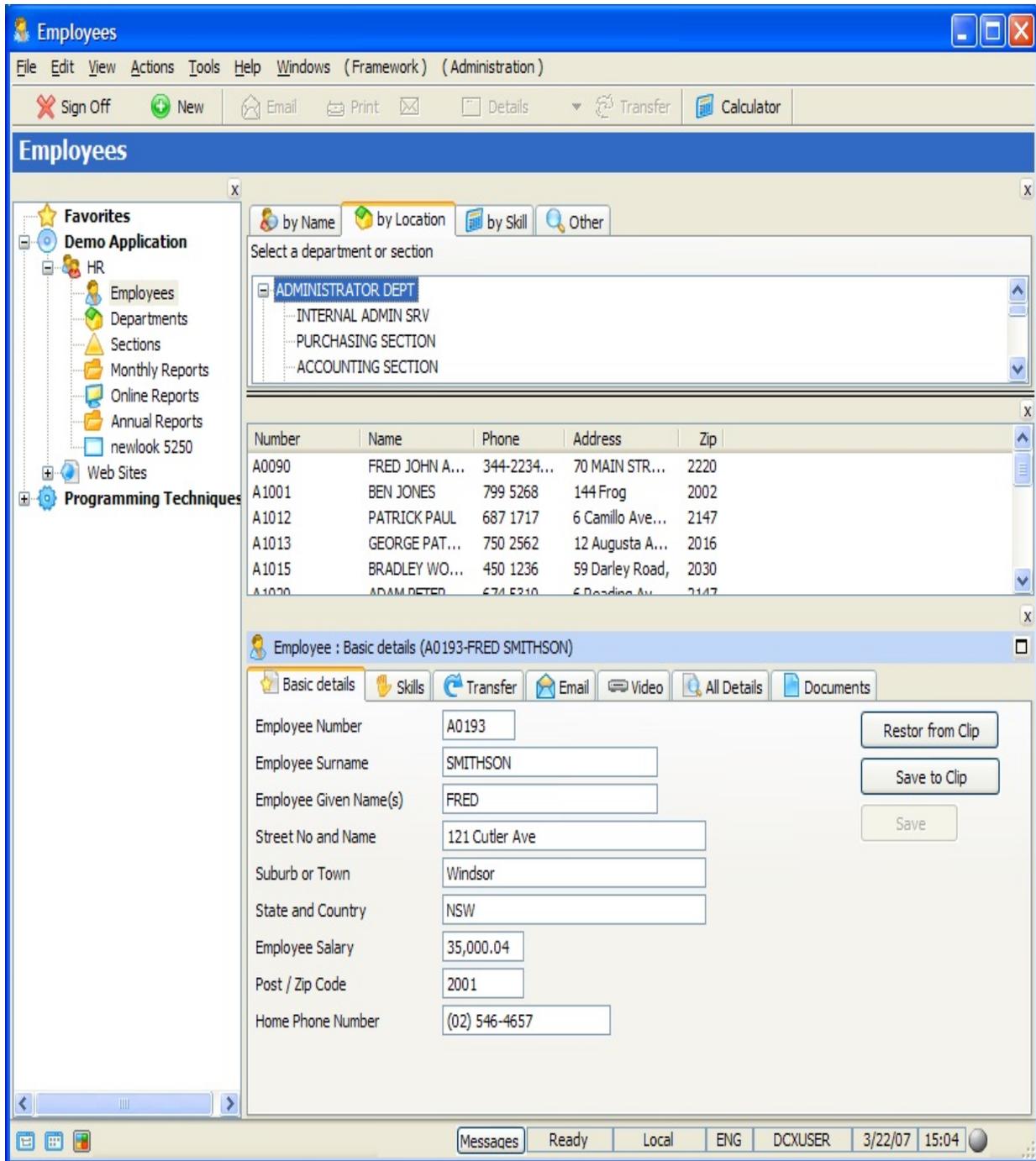
If the filter pane was positioned above the instance list pane like this:



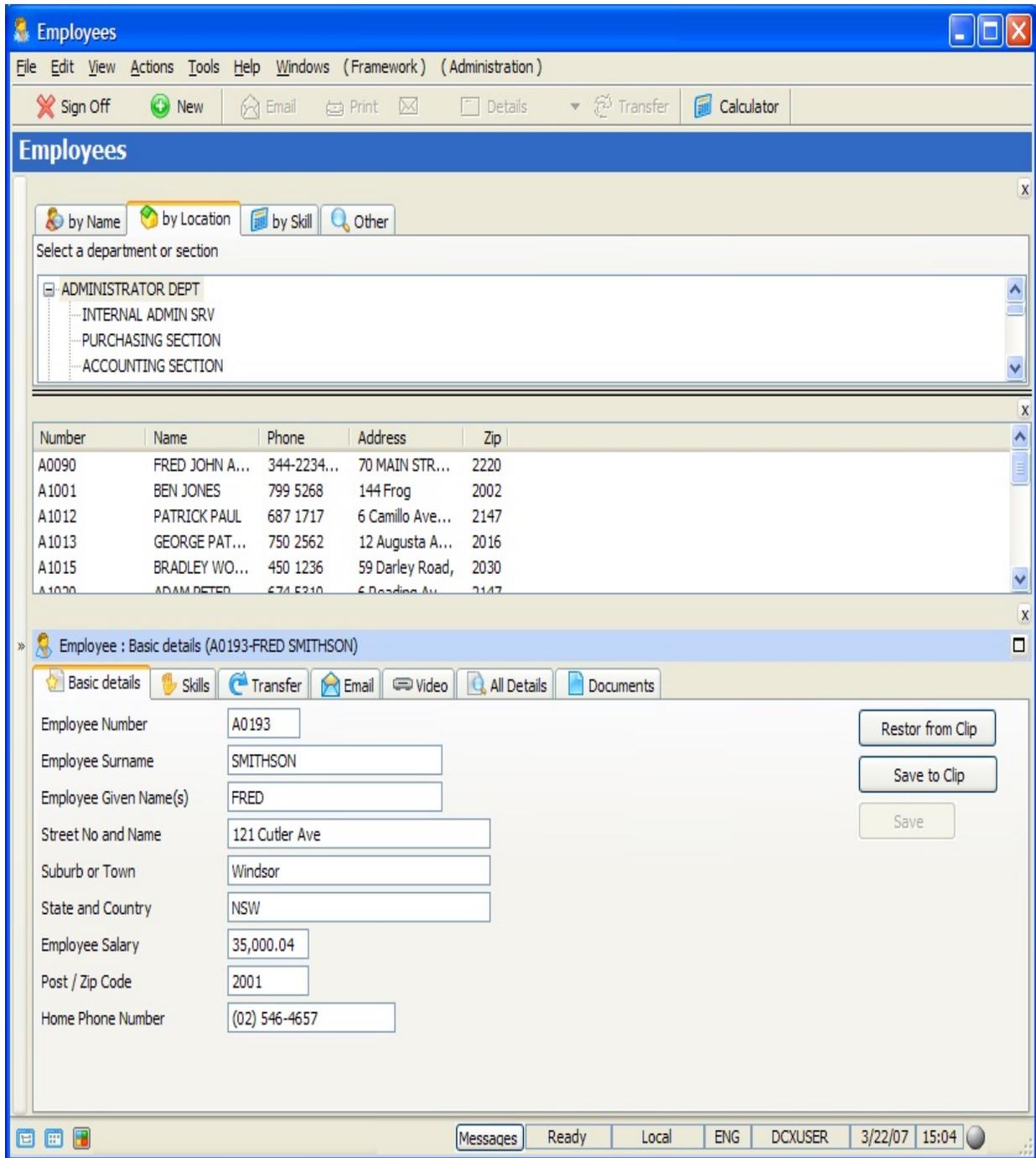
Then clicking the 'shrink' button causes it to shrink to a slim horizontal bar containing the words 'Expand Search filter for Employees'. Clicking on the bar expands the filter back into its normal position again when it needs to be used.



Commonly the shrink button is used to hide the navigation pane while working:



By clicking the navigation pane's shrink button it is reduced to a slim bar down the left hand edge of the form, yielding more application working space on the form:



In VLF.NET applications this option controls whether panes can be pinned and unpinned.

This property is in the [Framework Details](#) tab.

Allow Search/Recently Used Limit

Applies to **Windows** and **.NET** applications.

This option defines how many recently used business objects should be remembered. The minimum sensible value is 1 and the maximum is 100.

For more information see [Quick Find Box on the tool bar](#).

This property is in the [Framework Details](#) tab.

Allow Selection from Navigation Pane

Uncheck this option if you do not want this business object to appear in the navigation pane (see [Framework Window](#)). Typically this option is unchecked for business objects that should not be directly selectable by themselves (objects with a relationship type Child in Instance List/Relationships tab).

For example, an ORDER-ITEM business object may be defined in the Framework, but it is not something that would be directly selectable by a user. Normally an ORDER-ITEM would only be accessible via its parent business object ORDER.

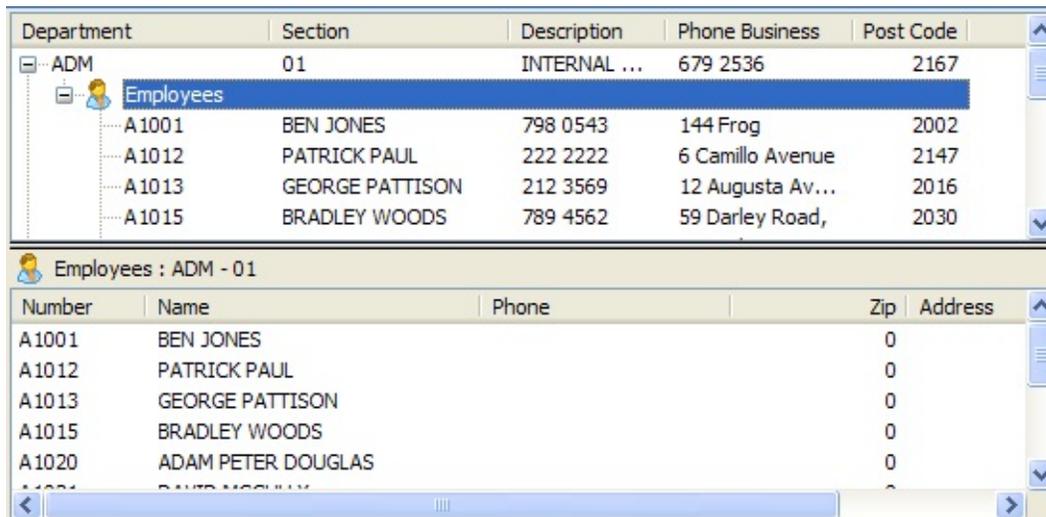
Also see [Work with Hidden Child Objects](#).

This property is in the [Identification](#) tab and the [Instance List/Relations](#) tab.

Allow Side by Side Display

Use this option to specify that a separate instance list is displayed for this child object to the side or below the main instance list..

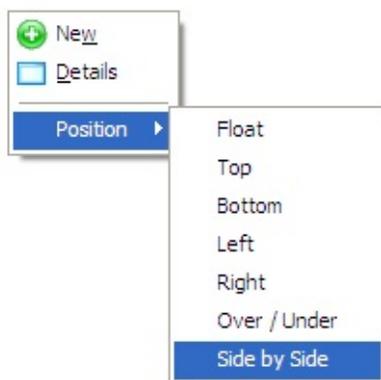
In this example the option has been selected. When you click on the Employees node in the tree, the list of associated employees appear under it like this:



Department	Section	Description	Phone Business	Post Code
ADM	01	INTERNAL ...	679 2536	2167
Employees				
A1001	BEN JONES	798 0543	144 Frog	2002
A1012	PATRICK PAUL	222 2222	6 Camillo Avenue	2147
A1013	GEORGE PATTISON	212 3569	12 Augusta Av...	2016
A1015	BRADLEY WOODS	789 4562	59 Darley Road,	2030

Number	Name	Phone	Zip	Address
A1001	BEN JONES		0	
A1012	PATRICK PAUL		0	
A1013	GEORGE PATTISON		0	
A1015	BRADLEY WOODS		0	
A1020	ADAM PETER DOUGLAS		0	
A1021	DAVID MCGILLIVRAY		0	

By default in side-by-side display the separate instance list actually appears underneath. To make it side by side, use the right mouse menu and then select Position and then Side by Side:



This is the type of display layout you will see:

Department	Section	Employees : ADM - 01		
ADM	01	Number	Name	Phone
Employees		A 1001	BEN JONES	
A 1001	BEN JONES	A 1012	PATRICK PAUL	
A 1012	PATRICK PAUL	A 1013	GEORGE PATTISON	
A 1013	GEORGE PATTISO	A 1015	BRADLEY WOODS	
A 1015	BRADLEY WOODS	A 1020	ADAM PETER DOUGLAS	
A 1020	ADAM PETER DOU	A 1021	DAVID MCCULLY	
A 1021	DAVID MCCULLY	A 1025	MARY ROBINSONNNN	
A 1025	MARY ROBINSONN	A 1027	ALAN MORRISON	
A 1027	ALAN MORRISON	A 1111	WARREN PETER VEREY	
A 1111	WARREN PETER V	A 1404	GILL MRS BRICK	
A 1404	GILL MRS BRICK	A 1509	ROBERT REDFORD	
A 1509	ROBERT REDFOR			
ADM	02			

The advantage of this type of display is that different children can have quite different sets of additional columns to their parent, or to each other.

The side by side column layout details come from the currently selected child business object.

This property is in the Business Object [Instance List/Relations](#) tab.

Allow this Object to be Opened in a New Window

Use this option to control whether this object can be opened in a separate window.

If an application is not allowed to be opened in a separate window, the views or business objects it contains cannot be opened in a separate window.

If the entire Framework can be opened in a secondary window, implicitly the user can access secondary instances of any application, view or business object within it by specifically navigating to them. So, to restrict access to individual business objects you should make sure that the option is also controlled at the Framework level.

This option is only applicable if the [Show the 'Windows' Menu in this Framework](#) option is selected in the Framework. This option only applies to **Windows**.

The options are:

- | | |
|---------------------------|---|
| Never | This object cannot be opened in another window by an end-user. |
| Manually | This object can be manually opened in another window by an end-user. They do this by using the Window -> Open in a New Window main or popup menu options. |
| Automatically | This object should be opened in a new window automatically when it is selected in the navigation pane. In effect this means that this object will always operate in a separate window to the one it is launched from. |
| Automatically or Manually | This object should be opened in a new window automatically when it is selected in the navigation pane, and it can be manually opened by an end-user. |

Note that none of these options have any impact on the ability of developers to programmatically open new windows by using the `avShowWindow` method (see [Programmatically Creating and Managing Windows](#)).

This property is in the [Identification](#) tab.

Allow User to Clear Instance List

Check this option if the generated code is to include an option that allows the end-user to clear the instance list.

By allowing end-users to selectively clear the instance lists (and by providing an array of different filters) the end-user can build instance lists that match quite complex business criteria.

E.g.: A list of all employees who work in the ADM or MKT departments, and all employees who started work in the first 3 months of this year, is easy using 2 very simple employee filters.

This is a [Program Coding Assistant](#) property.

Allow Users to Switch Views

Select this option to let the user choose how to display the [Application Bar](#).

This property is in the [Framework Details](#) tab.

Alter Development Status

Use this option to set a development status indicator and notes for the object.

See [Development Status Feature](#).

This property is in:

The [Commands Enabled](#) tab

The [Identification](#) tab

As the Default Command

Enable this command as the [Default Command](#).

When a command is enabled as the default command, it is executed immediately, when an application, business object or business object instance is selected.

You should only have one default command for any application or business object.

This property is in the Commands [Other Options](#) tab.

ASCII Translation Table Name and Library

This is the name and library of the System i translation table that should be used to perform translations to ASCII format.

The table is only used in System i Web browser applications and only during the sign on process.

The default value for this property is *JOB for RDMXL enabled partitions, otherwise it is QASCII. The default System i library name is QSYS.

*JOB means the translation table will be generated based on the client code page and the System i server job's CCSID.

This property is in the [Web/RAMP Details](#) tab.

Associated AJAX Function

AJAX functions are normal RDML or RDMLX functions which receive information from and return information to AJAX page(s).

See [Framework-AJAX Applications](#).

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

Automatic Save Time in Minutes

Specify here how often you want the Framework settings to be saved automatically.

When you are prototyping, it is helpful to have the settings saved fairly frequently, so that you do not lose information in case of a problem.

If you do not use the automatic save (by setting this value to zero), the settings are saved when you close the Framework.

This property is in the [Framework Details](#) tab.

Automatically Enable for New Applications

Select this option to automatically enable this command for all new applications.

This property is in the Commands [Other Options](#) tab.

Automatically Enable for New Business Objects

Select this option to automatically enable this command for all new business objects.

This property is in the Commands [Other Options](#) tab.

Automatically Enable for New Business Object Instances

Select this option to automatically enable this command for all new business object instances.

This property is in the Commands [Other Options](#) tab.

Automatically Increment when Saving

If this option is checked, the revision number will be automatically incremented when you save your Framework.

This property is in the Framework [Identification](#) tab.

Bitmap

Choose a bitmap from the list. The bitmap will be assigned to the currently selected object.

A bitmap for an application is displayed in the right-hand corner of the Framework window. The bitmap for a command is displayed in the toolbar button if it has one.

You may want to review [How do I enrol my own bitmaps and icons?](#)

This property is in the Framework [Help About](#) tab and the [Bitmaps and Icons](#) tab.

Blank Lines Around Comments

Specify the number of blank lines in generated code that you want to precede single comment lines or comment blocks. Typical values are 0, 1 or 2.

This is a [Program Coding Assistant](#) property.

Business Object Command

Select this option if you want the selected command to apply to the business object itself. Business object commands are displayed when you right-click the business object.

Whether a command is best implemented as applicable to the business object itself or its instances depends on the context.

For example, let's assume you have a business object Employee, with instances such as 'employee A1234 Veronica Brown'. To allow the user to enter new employees, you would create the command New. Logically this command applies to the business object Employee, so you would make it a business object command.

To view details of an individual employee (in other words an instance of the Employee business object) you could add a command Details. You need to make this command an instance command, because it will display details of the individual employees (such as phone number, salary etc.), not about the business object. So for Details you would select the Instance Command option.

This property is in the [Commands Enabled](#) tab.

Business Object List

This list shows all the business objects in the Framework. Use it to select objects to be related with the business object currently being edited.

The relationships can be peer-to-peer or child relationships.

For more information, see [Different Types of Objects in Instance List \(Sub objects\)](#).

This property is in the Business Object [Instance List/Relations](#) tab.

Bypass Locks

VLF.WIN only.

This option can be used to indicate that this command handler is not blocked from execution by any form of Framework or RAMP screen locking.

It is intended for use by simple framework level command handlers like calculators that are initiated by buttons on the button bar.

It's important to understand that when you allow a command handler to bypass locks, the command needs to be immediately accessible to the user when a Framework RAMP lock is in place (eg: on the button bar, on visible tab or on a right mouse menu).

When a lock is in place a user cannot navigate through Framework interactions to ultimately make a bypass locks command accessible - that would require prescient software.

This property is in the [Commands Enabled](#) tab.

Caption

Enter here the caption for this object. Caption is the visible name of the object.

How the caption is shown depends on the type of object:

- For an application it is the text displayed for the application under its icon in the [Application Bar](#).
- For a business object it is the text displayed under its icon in the [Business Object Bar](#).
- For a command it is the name of the command displayed in menus and in the command tab.
- For a filter it is the name of the filter displayed in the filter tab.
- For a language it specifies the caption of the language.
- For a custom property it specifies the caption to be associated with the custom property. This caption is shown to the administrator.

This property is in:

The [Identification](#) tab

The [User Details](#) tab

The [Web/RAMP Details](#) tab

Caption (Singular)

Enter here the caption of your business object in singular.

Depending on the context, the Framework displays the caption of the business object either in singular or plural (for example 'Customer' and 'Customers').

For instance the command handler title area shows the caption in singular. The plural form is used in the navigation area and in the instance list heading or sub-heading.

This property is in the [Identification](#) tab.

Caption with Accelerator

The captions for commands are presented in contexts that allow them to be used with accelerator keys. For example, you might specify a command with caption "Details" and the accelerated caption "De&tails" which would appear as "Details" and allow Alt-E to be used to invoke the command.

This property is in the [Identification](#) tab.

Certificate File (PFX)

Applies to **.NET** applications.

If you have certificate file to digitally sign your VLF.NET applications, specify its full path and name here, otherwise leave this field blank. The certificate file should contain a PKCS#12 (or Personal Information Exchange) certificate and the file would normally have .pfx extension.

The verify button may be used to confirm that the file specified exists and is accessible.

The verify button does not verify that the file is a valid certificate file.

This property is in the [Developer Preferences – Web Server](#) tab.

Certificate File Password

Applies to **.NET** applications.

If your certificate file is encrypted and requires a decryption password, specify the password here, otherwise leave this field blank.

Any password you specify here is stored in the developer's preferences XML document in an encrypted format.

This property is in the [Developer Preferences – Web Server](#) tab.

Change Password IIP

Applies to WEB and WIN applications.

This is the shipped Change Password Imbedded Interface Point (IIP) which handles requests to change a user's IBM i password.

The default function is named UFU0006 in process UF_SYSBR. It is shipped with source code. Read the comments in the source code to learn how to use it.

Note: the nominated function MUST be RDMLX enabled because this feature is only supported by RDMLX.

For more information see [Imbedded Interface Points \(IIPs\)](#).

This property is in the Framework [User Administration Settings](#) tab.

Check Password Expiry

Check this box to compare the password's expiry date with the current date. Specify the number of days before the expiry date to start issuing warnings in Warn before (days).

This property is in the Framework [User Administration Settings](#) tab.

Check Still Connected Before Doing Database IO

Select this option if you want to include code that checks the connection status before any database I/O commands are executed.

This is a [Program Coding Assistant](#) property.

Client CCSID

Type in a 5 character long valid IBM i CCSID.

Remarks:

- This value is only valid for WEBEVENT transactions.
- The value must be exactly 5 characters long. Prefill with zeroes if required.
- WAM components always use 01208 (UTF-8).
- When in doubt always try using 00000.

This property is in the [Web/RAMP Details](#) tab.

Client Server Translation Table

For System i servers only, specify the client to server translation table to be used.

Note that these values are automatically defaulted from the client partition definition.

The default translation table used for Framework server definitions using RDMLX partitions/connections is *JOB.

Ensure that your client and server partition definitions match.

This property is in the [Server Details](#) tab.

Close Confirmation Message

Enter text for a confirmation message for end-users closing the VLF. The message will be shown before the VLF is closed.

If you do not want a confirmation message to be shown, leave this field blank.

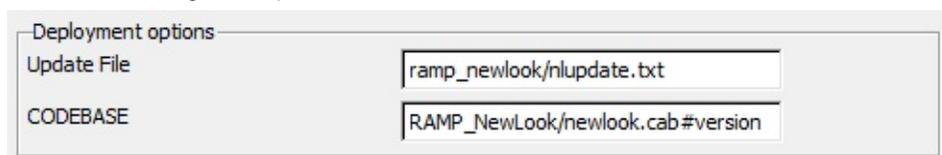
This property is in the [Web/RAMP Details](#) tab.

Codebase

Use this property to add a reference to the newlook.cab file on your webserver for when the application is deployed. This file hosts the newlook ActiveX control (nlocx.dll) and the update program (nlupdate.exe).

Internet Explorer will use the newlook.cab file to download and install the newlook ActiveX control if it is not already on the user's PC.

References to the newlook.cab file must specify the subdirectory in which it resides on your web server (which is below the directory containing the VF_SY120.js file).



The screenshot shows a dialog box titled "Deployment options" with two input fields. The "Update File" field contains the text "ramp_newlook/nlupdate.txt". The "CODEBASE" field contains the text "RAMP_NewLook/newlook.cab#version".

Alternatively, you can specify this value as a URL parameter when starting your application:

```
+NLCODEBASE=
```

This property is in the [Server Details](#) tab.

Code Table Data

The read-only grid shows the data in the table. To edit or add table data, click on the row you want to edit or add and then edit the values in the right hand panel.

To save the values you have edited, either click on a different row or press the Save button. If any validation errors result from the save they will be displayed in the Framework status bar.

You cannot edit key values once the row has been successfully saved.

If the table is a read only table you cannot edit the values in the right hand panel.

Save Button

The Save button saves the selected row.

It passes the data for the current row to the table data handler function, with the operation code of INSUPD - see the source for function UF_SYSBR/UFU0011 for an example of how the table data handler does the save.

Delete Button

The Delete deletes the selected row.

It passes the data for the current row to the table data handler function, with the operation code of DELETE - see the source for function UF_SYSBR/UFU0011 for an example of how the table data handler does the save.

Replace All Button

The Replace All button deletes all existing table data, and then saves all the data displayed.

It passes the operation code of DELETEALL to the table data handler function, which deletes all table data for the function, and then passes the data for each row to the table data handler function with the operation code of INSUPD - see the source for function UF_SYSBR/UFU0011 for an example of how the table data handler handles DELETEALL and INSUPD.

This property is in the Code Table [Data](#) tab.

Code Table Definition/ Reusable Part Data Handler (ID)

Enter the name (identifier) of the Reusable Part Code Table Data Handler.

This property is only relevant when the Use a Reusable Part check box is checked. The Framework will use this LANSAs component as the Code Table's data handler. See UF_TDH01 – Default Table Data Handler for an example.

This property is in the Code Table [Definition](#) tab.

Code Table Definition/ Use a Reusable Part

Check this box to use a reusable part as the VLF Code Table Data Handler. If this option is checked, at run time the Framework will look for a reusable part when expanding an instance list entry.

If this option is unchecked, the Framework will look for a function.

The reusable part must have ancestor #VF_AC024.

The component should redefine the uLoadTableData and uSaveTable methods as a minimum. See UF_TDH01 – Default Table Data Handler for an example.

This property is in the Code Table [Definition](#) tab.

Code Table Field Definitions

This contains the field definitions for the code table. When editing field data, there will be a column for every field defined in this grid.

Each entry in this grid can have the following options:

Field Name	<p>The choice for this value depends on the kind of Table Data Handler you are using.</p> <p>For Tables with data sourced from user defined files it is convenient to give the Table field the same name as the file field. For Tables with data sourced from the generic storage file it does not matter as long as the name is unique within this table and is non-blank.</p>
Key	<p>This specifies this field as an identifying key field for the table. Its key number (e.g. whether its AKey1 or AKey2) is determined by its sequence in the grid, relative to the other key fields.</p> <p>There can be up to 5 keys, and they can be either Numeric or Alphanumeric. The key fields must be chosen so that they uniquely identify a table data row.</p> <p>Alphanumeric keys can be up to 32 characters long.</p> <p>Numeric keys must fit within a 15,5 decimal field.</p>
Caption	<p>The field caption is used to identify the field when editing data or selecting a field.</p>
Type:	<p>The field type - Alpha or numeric</p>
Max Length	<p>The maximum length a value for this field can be.</p> <p>Note: Alpha key fields should never be more than 32 characters.</p> <p>Numeric key fields should fit within a field of 15, 5</p>
Decimals	<p>The number of decimal places for numeric fields.</p>
Upper	<p>Values in this field should always be upper case</p>

Case
Only

Move Up

The button changes the sequence of a field definition.

The relative displayed sequence determines the key sequence for key fields

Move Down

The button changes the sequence of a field definition.

The relative displayed sequence determines the key sequence for key fields

Delete

The button deletes a field definition from this Table

This property is in the Code Table [Definition](#) tab.

Columns for Instance Lists

This grid allows you to specify column details used by the standard shipped instance lists.

If you have created your own snap-in instance list (an option for Windows applications), then none of the details specified in this grid apply.

Use these values to control how instance list values are displayed:

- **Column Sequence:** Zero here means that this column will not be displayed. Any other value determines where this column will be displayed relative to other displayed columns from left to right.
- **Column Type:** This is the identifier that the Framework uses to determine which column to put a value into when it extracts information from the instance list. Additional columns must be used contiguously within type (i.e.: you can't use additional alphanumeric columns 1, 4 and 5 in you programs. You must use 1,2 and 3, even if you choose to display just columns 1 and 3).
- **Column Caption:** This is the heading that will be shown at the top of this column on the instance list.
- **Column Width:** This is the percentage width that this column will have when displayed in the instance list. A zero value means use the remainder of the instance list. If you don't want a column to appear, set its Column Sequence to zero.
- **Column Decimals:** This is the number of decimal places that will be shown, for columns of type additional numeric.
- **Edit Code:** Edit codes can be applied to numeric additional columns. To set an edit code for a numeric value, select the edit code from the dropdown on the instance list definition grid. The edit code identifier appears first followed by an example of the resulting display format. The edit codes available map to the standard LANSAs codes 1,2,3,4,A,B,C,D,J,K,L,M,N,O,P,Q,Y and Z. Default is no edit code is applied. Note that:
 - **This feature is for VLF-WIN only.**
 - Perfect real or implied visual decimal point alignment between positive and negative values may not be achievable. Depending upon the font being used the visual impact may be better or worse.
 - Values sent to MS-Excel may not be interpreted as numeric values because trailing CR, “-“ or breaking space pad characters (X0A) may not be automatically considered to be numeric content by MS-Excel.

Date/Time Output Format: Select the output format for Date/Time additional columns. The available formats are for both Date and DateTime values and correspond to all the formats available for the AsDisplayString intrinsic function for both RDMLX Date and DateTime fields. The default value is eight character system format, SYSFMT8. This property is only applied to Date/Time additional columns.

UTC Conversion: Change this property if a Coordinated Universal Time (UTC) conversion is required for the DATETIME value shown on the instance list. The default value, Local -> Local, represents no conversion. The value Local -> UTC assumes that the value added to the instance list is local time and removes the UTC offset from that value and displays the converted DATETIME value (now UTC) on the instance list. Similarly, the value UTC -> Local assumes a UTC value was added to the instance list and converts that to local time. This property is only applied to Date/Time additional columns with DATETIME values.

For example, the Employees business object has a Visual ID1 of the employee name, a Visual ID 2 of the employee code/ID and additional alphanumeric and numeric columns defined. You might enter details into the grid like this:

Sequence	Type	Caption	Width % (Total 100%)	Decimals	Edit Code	Date/Time Output Format	UTC Conversion
10	VISUALID1	Name	20		Default	SYSFMT8	Local -> Local
20	VISUALID2	Code / Id	10		Default	SYSFMT8	Local -> Local
30	ACOLUMN1	Address Line 1	10		Default	SYSFMT8	Local -> Local
40	ACOLUMN2	Address Line 2	10		Default	SYSFMT8	Local -> Local
50	ACOLUMN3	Address Line 3	10		Default	SYSFMT8	Local -> Local
70	NCOLUMN1	Zip Code	10		Default	SYSFMT8	Local -> Local
75	ACOLUMN4	Business Phone	10		Default	SYSFMT8	Local -> Local

This causes the shipped instance list to display an instance list like this (web or Windows)

Name	Address Line 1	Address Line 2	Address Line 3	Zip Code	Business Phone	Department	Section
BLOGGS,FRED JOHN ALAN	70 MAIN STREET	NEWTOWN NSW	AUSTRALIA	2220	654 6475 X432	FLT	03
SMITHSON,FRED	121 Cutler Ave	Windsor	NSW	2034	(02) 354-5647	ADM	05
MISS SIMPSON,ANNE	33 anne street	anneville	annes	2145	090909	AUD	03
JONES,BEN	144 Frog	PYMBLE.	NSW.	2001	798 0543	ADM	01
SMYTHE,JOHN	20 Cobbitty Ave...	WERRINGTON.	NSW.	2100	798 4381	ADM	02
SMITHE,Robert	29 Arthur Road,	DEE WHY.	NSW.	2000	406 6395	FLT	02
SMITHSON,PAUL	41 William Road,	GRANVILLE.	NSW.	2144	239 9174	ADM	03
SMITHS,PETER	72 Mullane Avenue,	BAULKHAM HILLS.	NSW.	2147	777 7265	ADM	02
SMITHERS,JACK	8 Croydon Road,	CROYDON.	NSW.	2050	32 1667	TRVL	03
SNELL,GEORGE	6 Anthony Avenue,	PADSTOW.	NSW.	2164	44 2965	AUD	01
SNEDDON,ALLAN	22 Railway Parade,	KOGARAH.	NSW.	2160	630 8888	AUD	01

Note that at run time, for Windows applications only, the end-user may resize

columns or even change the order of columns (using drag and drop), and the Framework will remember these changes. So these settings are just a starting point, for Windows Frameworks.

Windows applications also have the option of using a snap in instance list, instead of the shipped instance list. To add more features to your displayed instance list you should investigate using your own snap-in instance list. For more details, refer to [Optionally Create your own Instance List](#).

This property is in the Business Object [Instance List/Relations](#) tab.

Command Tab Location

Applies to **Windows** and **.NET** applications.

Use this option to specify how the tabs of a command tab folder are displayed for this object. The tabs can be displayed on the top, bottom, left or right of the tab folder.

If the [Command Tab Style](#) is Buttons or FlatButtons, the only possible location is top.

This option only applies to **Windows** Framework applications.

This property is in the [Command Display](#) tab.

Command Tab Show All

Use this option to control whether the tabs for all commands of an object are displayed when the user selects one of its commands.

Automatic This value sets the following behavior:

- For business object instances, the tabs for all commands are displayed when the user selects one command.
- For the Framework, applications and business objects, only the tab for the selected command is displayed.

True When the user selects a command for this object, tabs for all commands that apply to it are displayed.

False When the user selects a command for this object, only the tab for the selected command is displayed.

Note: This option is only applicable to **Windows** applications. Web browser applications ignore this setting and behave as if "True" was specified.

You can specify this property for:

- The command tab of the Framework.
- The command tab of an application.
- The command tab for a business object and the command tab of its instances.

This property is in the [Command Display](#) tab.

Command Tab Style

Applies to **Windows** and **.NET** applications.

Use this option to specify how the command tab folder for this object will be displayed. Tabs is the normal tab sheet display, Buttons and Flat Buttons display the tabs as buttons. Stacked displays the Tab folder as a set of vertical bars. Clicking on a bar displays the desired tab sheet. Stacked is not applicable to VLF.NET applications.

Buttons, FlatButtons and Stacked can only be used with [Command Tab Location](#) Top.

This property is in the [Command Display](#) tab.

Commitment Control

Select this option to use commitment control.

This property is in the [Server Details](#) tab.

Compile Framework as Microsoft .NET 2.0 Executable

Do not check this option unless requested to by your LANSA product vendor.

This property is in the [Framework Details](#) tab.

Component Identifier

The identifier of the LANSAs component used as filter or command handler.

See [Component Names and Identifiers](#).

This property is in the [Commands Enabled](#) tab the [Filter Snap-in Settings](#) tab.

Contains Favorites

Use this property to enable shortcuts to an end-user's favorite business objects to be stored in this application. When this property is selected, the end-users simply drag business objects they commonly use into the application to create the shortcut.

The actual business objects remain in the application they belong to.

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

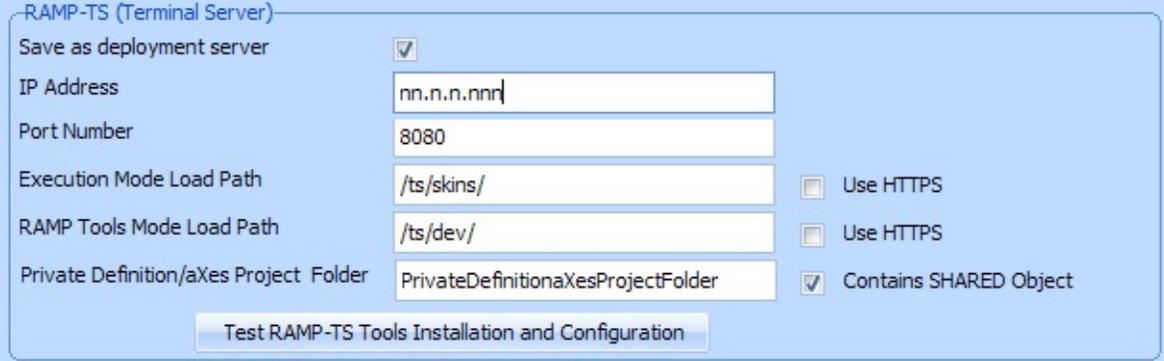
This feature is available in Windows only, so the Allow on Web property is normally unchecked if Contains Favorites is checked. Favorites applications should not contain any Application Views or real business objects.

An application designer may choose to create multiple favorites applications (for example Normal Favorites, End of Month Favorites and Year End Favorites).

This property is in the [Identification](#) tab.

Contains SHARED Object

Check this option to indicate to RAMP-TS that the SHARED Object file is to be retrieved from the nominated Private Definition/aXes Project Folder.



The screenshot shows a configuration window titled "RAMP-TS (Terminal Server)". It contains several fields and checkboxes:

- Save as deployment server:**
- IP Address:**
- Port Number:**
- Execution Mode Load Path:** Use HTTPS
- RAMP Tools Mode Load Path:** Use HTTPS
- Private Definition/aXes Project Folder:** Contains SHARED Object

At the bottom of the window is a button labeled "Test RAMP-TS Tools Installation and Configuration".

A RAMP-TS session's SHARED script object is defined in the script file UF_SY420_RTS.JS and by default resides in folder \aXes\ts\skins, which means it is common to all RAMP sessions within the Axes instance.

If you check this option you are indicating that you will use a private version of this file and the SHARED Object will only apply to this particular RAMP-TS session. You will need to copy the file to your private definition folder.

This also means the SHARED object file will reside in folder \aXes\ts\screens\
<your private definition/axes project folder>.

This property is in the [Server Details](#) tab.

Copyright Notice to Be Used

Specify the copyright notice you would like to use in your generated code.

For example : © Acme Software, 2014. All rights reserved.

This is a [Program Coding Assistant](#) property.

Copyright Text

Enter here the copyright text.

This property is in the [Help About](#) tab.

Create Component

You can use this field to specify the name of the component to be automatically created provided you started the Framework from within the Visual LANSAs Development environment (IDE) using the Tools tab on the ribbon.

The type of component created depends on the mode you are running the Program Coding Assistant in:

Mode	Component Type
-------------	-----------------------

Windows	Reusable Part
---------	---------------

WAM	A Web Application Module
-----	--------------------------

The names given to the components must adhere to LANSAs naming rules. If you want to use different rules, see the instructions for method routine `avValidateLANSAName` in `UF_SYSTEM`.

The names must not already be in use, except for the Process name which may exist already. By default the process name can be maximum 8 characters.

The Create button will be enabled if valid component names are entered AND the Framework was started via the Visual LANSAs Development environment. It may take the Framework several seconds to determine this.

The Create button is optional. You can still copy and paste the source code into a component you create yourself.

This is a [Program Coding Assistant](#) property.

Database Name

Enter here the name for the Windows or UNIX database.

This property is in the [Server Details](#) tab.

Database Password

For Windows or UNIX databases. Enter here the database user password if the password is different to the user's normal sign on user password.

For example the default password for an SQL Anywhere database is SQL.

This property is in the [Server Details](#) tab.

Database Type

For Windows or UNIX databases. Select the type of other database.

This property is in the [Server Details](#) tab.

Database User

User profile for Windows or UNIX servers. This user profile will be used as the database user for all users connecting to the server.

For example, the default user profile for an SQL Anywhere database is DBA.

If a database user is not specified for a Windows or UNIX server, the system user profile is used as the database user profile when connecting to the server.

For more complex connections the user has the option of specifying their own connect logic in UF_SYSTEM, method avPrivateConnect.

This property is in the [Server Details](#) tab.

DBCS Capable

For servers, select this option if the server is DBCS capable.

For languages, check this option if the language you are defining is a DBCS language. Normally this value is defaulted for you and you should use the default value unless instructed otherwise by your product vendor.

This property is in the [Server Details](#) tab and the [Web/RAMP Details](#) tab.

Default Command

- Yes** The selected command is the default command for this object.
For instance commands, a default command is executed when the user clicks or double-clicks on an entry in the instance list. If the command handler is shown as a tab sheet, it is presented leftmost in the folder, regardless of how you may have sequenced any other tab sheets within the folder.
- No** The selected command is a normal command.
Normal instance commands are not displayed by default when the user clicks on an instance list entry unless the user has chosen this command when they clicked on the previous instance list entry.
- Never** Applicable to instance commands only.
This command is never the default. Even if the user chose this command for the previous instance list entry, when the user clicks on the next entry this command is not shown.
When subtypes are defined for a business object and an instance command is not enabled for all subtypes, the Default Command option for the command should be defined as Never.

The concept of a default command at the Framework level has no meaning. If you want to execute a specific command when a Framework starts, refer to [New UF_SYSTM IIP \(Imbedded Interface Point\) methods that you can override \(Windows\)](#) and the SwitchCommand parameter in [Web Application Start Options \(web\)](#).

This property is in the [Commands Enabled](#) tab.

Default Font when Printing a Report Using Windows

Specify here the default font for printing reports. This has to be a font known to Windows.

This option only applies to **Windows** Framework applications.

This property is in the [Framework Details](#) tab.

Default Properties for Fields on Filter Panels

Specify the default DEFINE_COM values that should be added to fields inserted onto panels when generating filters.

The shipped default value for this setting is: LabelPosition(Top) Marginleft(0) Margintop(19) Height(38).

This is a [Program Coding Assistant](#) property.

Default Properties for Fields on Handler Panels

Specify the default DEFINE_COM values that should be added to fields inserted onto the header area of panels when generating command handlers.

The shipped default value for this setting is: LabelPosition(Left).

This is a [Program Coding Assistant](#) property.

Defined In

Displays the caption of the Framework, application or business object in which the custom property is defined. When reviewing custom properties at one level (eg: business object level) the custom properties from higher levels will also be displayed (eg: from the Framework level) for reference, but they cannot be changed. A message to this effect appears at the bottom of the form.

This property is in the [Custom Properties](#) tab.

Displayed Field

Use this option to select which of the table fields is to be the one shown for each combo box entry, or as the caption for each radio button.

This property is in the Code Table [Visualization](#) tab.

Development Status Captions

You can modify the captions indicating the status of development.

For more information, see [Development Status Feature](#).

This property is in the [Framework Details](#) tab.

Developer Preferences XML File

Use the Server Settings XML File property to specify the name of the file where developer preferences for the Framework are stored. This file is located in the Execute directory of the current partition.

This property is in the [Framework Details](#) tab.

Divert Locks

Select this option to store object locks on the server.

If this option is selected and a connected workstation locks an object, the lock is stored on the server (diverted to the server) and will stop other workstations from working with the object.

If this option is not selected, the lock is stored locally on the workstation and only applies to that object on that workstation.

This property is in the [Server Details](#) tab.

Double Click for Default Command

Applies to **Windows** only.

Select this option to invoke the default command for a business object instance when the user double-clicks it.

This property is in the Business Object [Instance List/Relations](#) tab.

Do you want to use a mapped drive when uploading to the webserver?

If the webserver is on a different machine to your development machine, there needs to be a way to get files from the development machine to the webserver.

The easiest way to do this is to map a drive on the development machine that points to the webserver. (On the development machine, use windows explorer --> Tools --> Map Network Drive)

If you have a mapped drive that points to the webserver, specify it here.

If you don't have a mapped drive you will need to manually copy your framework files to the images directory, and also to the project folder, on the web server. (every time you save changes to the framework)

Or, you could investigate using an ftp script - see (framework) --> (framework properties) --> developer preferences --> Script for uploading to your LANSA for the web folders

See [Script for Uploading to your LANSA for the Web Folders](#).

This is a [Web Configuration Assistant](#) property.

Drag and Drop the Fields

Choose which fields you want to display in the list of child objects, and which fields you want to edit.

This is a [Program Coding Assistant](#) property.

EBCDIC Translation Table Name and Library

This is the name and library of the System i translation table that should be used to perform translations to EBCDIC format.

The table is only used in System i Web browser applications and only during the sign on process.

The default value for this property is *JOB for RDMXL enabled partitions, otherwise it is QEBCDIC. The default System i library name is QSYS.

*JOB means the translation table will be generated based on the client code page and the System i server job's CCSID.

This property is in the [Web/RAMP Details](#) tab.

Edit Panel

Drag here the child object fields you want to edit.

This is a [Program Coding Assistant](#) property.

Email Address

The email address of the user.

This property is in the [User Details](#) tab.

Email Zip File To

Specify here the email address of the recipient of the xml files containing the definitions of your prototype. The recipient can extract the contents of the emailed zip file to their partition execute directory, and then run the Framework to see what the application prototype looks like.

This property is in the Framework [Export Design](#) tab.

Enable Child when Parent Selected

Default is checked.

When this option is checked, this indicates that when an instance of the parent object (eg: SECTIONS) is selected, all object level commands associated with the child object (eg: EMPLOYEES) should be enabled and made available for use. If this option is unchecked the child's object level commands are not enabled.

This option only applies when the business object is not selectable from the navigation pane. This option is disabled if the 'Allow Selection from navigation Pane' option is checked.

This property is in the Business Object [Instance List/Relations](#) tab.

Enable Clear List Button

Select this option to enable the Clear List button in the [Instance List](#).

When creating your filters you need to decide whether a new search will automatically clear the contents of the instance list. If you clear the list automatically, there is no need for the Clear List button.

If you decide not to clear the list automatically, the user has the option of using multiple searches and different search criteria to build up the instance list. In this case you would enable the Clear List button so that the users can clear the list when required.

This option is only applicable when using the standard shipped instance lists. If you are using your own snap-in instance list you can enable the clear list capability by whatever means you feel is appropriate.

This property is in the Business Object [Instance List/Relations](#) tab.

Enable Command

Drag a command to the Enabled list to enable it for your object.

When you have enabled a command, you can specify how it is displayed and what its command handler is.

This property is in the [Commands Enabled](#) tab.

Enable Development Status Feature

Enables the [Development Status Feature](#) which allows developers to attach a status and notes to framework objects.

Note: These are only visible when

- The Framework is run in development mode
- This property is enabled
- The Framework is run using Render Style M.

This property is in the [Framework Details](#) tab.

Enable Framework for AJAX style applications

You can now create AJAX style command handlers for Framework web applications for optimal speed and functionality.

See [Framework-AJAX Applications](#).

This property is in the [Framework Details](#) tab.

Enable Framework for WAMS

All Frameworks support the design and execution of native Microsoft Windows applications.

Check this option if you also want your Framework to support the design and execution WAM browser applications.

After changing this option you should always save and restart your Framework.

See also:

[Computer System Requirements](#)

and [Other Requirements](#).

[Should you use Windows or Web Browser Applications?](#)

This property is in the [Framework Details](#) tab.

Enable Framework for Web browser Applications

All Frameworks support the design and execution of native Microsoft Windows applications.

Check this option if you also want your Framework to support the design and execution Web browser applications.

After changing this option you should always save and restart your Framework.

This property is in the [Framework Details](#) tab.

Enable Framework for WEBEVENT Functions

Check this option only when you have an existing Framework that already contains WEBEVENT filters or command handlers and you wish to add more.

Otherwise always uncheck this option.

The ability to use WEBEVENT filters or command handlers is a deprecated feature.

Avoid creating new WEBEVENT filters or command handlers.

This property is in the [Framework Details](#) tab.

Enable Parent when Child Selected

Default is checked.

When this option is checked, this indicates that when an instance of the child object (eg: EMPLOYEES) is selected, all the commands associated with the parent object (eg: SECTIONS), should also be enabled and made available for use. If this option is unchecked the parent's commands are disabled.

This option only applies when the business object is not selectable from the navigation pane. This option is disabled if the 'Allow Selection from navigation Pane' option is checked.

This property is in the Business Object [Instance List/Relations](#) tab.

Enable Peers when Selected

Indicates whether any appropriate peer object(s) command(s) should be enabled when this business object is selected.

Default is true (checked).

This property is in the Business Object [Instance List/Relations](#) tab.

Enable Popup Panels

If checked, this enables instance list pop up panels for any instance list entries that have a pop up panel defined for their business object, provided the end user has not disabled instance list pop ups. (The end-user can do this by right-mouse clicking on the instance list and disabling Instance List Pop Ups.)

This feature is only available for the primary instance list for frameworks being run in Mixed mode (Render mode M).

This property is in the Business Object [Instance List/Relations](#) tab.

Enable the Position Menu Option

In Framework Windows applications:

Select this option to disable the Position menu option on pop-up (right-click) menus.

A designer can choose to prevent users from changing the relative position of the Frameworks display panes. When this option is disabled, the user will not be able to cause a pane to float either.

In .NET applications:

Select this option to allow panes to be docked, undocked, and dragged around.

This property is in the [Framework Details](#) tab.

Encrypt XML Files

Use this option to indicate whether the XML files used to store the framework definition, server definitions, table definitions and user definitions should be stored in an encrypted format.

Note: The storing of user definitions in XML files is an old feature that is rarely used. You should not use this feature in new frameworks.

When this option is enabled the content of the <name>.XML files is encrypted as they are saved and versioned (if versioning is enabled). An encrypted <name>.XML.UNENC copy of the file is also created and versioned if versioning is enabled.

You must always keep the <name>.XML.UNENC file(s) because they are in plain text format and protect you from significant data loss should your encryption key ever be lost or an encrypted file be accidentally corrupted.

You should deploy the <name>.XML encrypted file versions only. Since the content is encrypted it is very difficult for anyone to modify the content at a deployed location.

The encryption key is composed of two 8 character keys. One is supplied by you and the other by the VLF - making it very difficult for either party to externally decrypt the file content.

Your part of the key is always provided by your versions of the UF_EXEC, UF_DESGN, UF_ADMIN and UF_DEVEL system entry points. Refer to the shipped example source code for more details of how to set up your own encryption key. If you use this option and do not create your own system entry points, the default 8-byte user encryption key of "UDEFAULT" is used. Note that changing the encryption key in the system entry point by itself will not enable XML file encryption – you also need to enable this framework option.

If your encrypted XML file becomes unusable for some reason or if you lose your encryption key, delete the <name>.XML files that are encrypted and rename the <name>.XML.UNENC versions back to <name>.XML and then restart the framework as a designer.

This property is in the [Framework Details](#) tab.

End User can change theme

The Framework theme is specified when the Framework is designed, but you can also specify that the end-user can override the theme. If you select this option, end-users can override the visual style for their application using the Overall Theme option of the Windows menu.

Note that it may be necessary on some PCs for the user to shut down and restart the Framework before the theme is fully implemented.

This property is in the [Visual Styles](#) tab.

End-users Must Sign on to this Framework

You can specify that end-users must sign on to use the Framework in these situations:

- Never
- In both Windows and Web browser applications
- In Web browser applications only
- In Windows applications only

Note that disabling a sign on in an environment is not the same as disabling user authority (see [Use Framework Users and Authority](#)).

When a sign on is disabled but user authority is enabled, then the user profile is obtained from the operating environment and/or any authority related imbedded interface points (IIPs) that you have supplied.

The user profile is then subject to the normal authority rules within the Framework.

Note that you can use these options even if the [Use Framework Users and Authority](#) option is not selected.

This property is in the Framework [User Administration Settings](#) tab.

Engine

Use this setting to change the RAMP-TS emulation engine. Setting this property to aXes-TS2 will enable the use of aXes-TS2 compatible browsers for RAMP-TS applications.

See the aXes guide for details on the aXes-TS2 emulation engine.

This property is in the [Server Details](#) tab.

Execute as Hidden Command

Check this option to run a command without any user interface, that is, hidden from the user. The option applies to both Windows and Web Framework applications.

Hidden commands should always have the Default Command option set to NEVER for instance commands or NO for business object commands and the Hide All Other Command Tabs option should never be used.

Obviously, never attach RAMP Destination screens to hidden commands.

Also see [Hidden Command Handler Anatomy](#).

This property is in the [Commands Enabled](#) tab.

Execution Priority

Adjust this property to set a different execution priority for super-server jobs in the server definition. The default priority is 20 however if it is set to a lower figure the super-server job will have a higher priority and receive better service from the CPU. Conversely, a higher figure will give a lower priority to the super-server job. Note that this setting only affects VLF super-server connections and not any 5250 RAMP sessions.

See [Server Overrides](#).

This property is in the [Server Details](#) tab.

Execution Mode Load Path

Specifies the path that any RAMP-TS session started should use at runtime. You should consult with your product vendor before changing this value.

This property is in the [Server Details](#) tab.

Export Developer Preferences

Select this option to export all your current web server [Developer Preferences – Web Server](#).

This property is in the Framework [Export Design](#) tab.

Export Tables

Select this option to export code tables.

This property is in the Framework [Export Design](#) tab.

Export Framework Design

Select this option to export the Framework design.

This property is in the Framework [Export Design](#) tab.

Export Include All Versions

Select this option to export all the versions of the Framework.

This property is in the Framework [Export Design](#) tab.

Export Images Palette

Select this option to export the images palette.

This property is in the Framework [Export Design](#) tab.

Export RAMP Definitions

Select this option to export RAMP definitions.

This property is in the Framework [Export Design](#) tab.

Export RAD-PAD Notes

Select this option to export the RAD-PAD notes.

This property is in the Framework [Export Design](#) tab.

Export RAD-PAD Defaults

Select this option to export RAD-PAD defaults.

This property is in the Framework [Export Design](#) tab.

Export Servers

Select this option to export server definitions.

This property is in the Framework [Export Design](#) tab.

Export Users

Select this option to export user definitions.

This option is only enabled if your system Framework is configured to store user details in an XML file.

This property is in the User [Authorities](#) tab.

This property is in the Framework [Export Design](#) tab.

Field Prefix to be Used

Specify the character used on your system to denote a field or variable in generated code. The most typical value is the # sign.

This is a [Program Coding Assistant](#) property.

Fields That You Want to Appear in a List at The Bottom of Your Handler

See [Fields That You Want to Appear on the Top of Your Command Handler](#).

This is a [Program Coding Assistant](#) property.

Fields That You Want to Appear on the Top of Your Command Handler

The code generator can attempt to create a starting point command handler layout for you (you are expected to add to and change this layout later).

Visually, the starting point command handler is laid out like this example:

The screenshot shows a window titled "MJD_CH_01 - Test Command Handler Base". The top section, labeled "Header Area at Top", contains four form fields: "Employee Number" (value: ABCDE), "Employee Surname" (value: ABCDEFGHIJKLMNOPQRST), "Employee Given Name(s)" (value: ABCDEFGHIJKLMNOPQRST), and "Employee Salary" (value: 123,456,789.12). A "Save" button is located to the right of these fields. Below the header is a table labeled "List Area at Bottom". The table has four columns: "Skill Code", "Skill Full Description", "Ide Obtained for", and "Skill Acquired (DI)". The first row contains the values: ABCDEFGHIJ, ABCDEFGHIJKLMNOPQRST, A, and 12/34/56. The table is scrollable.

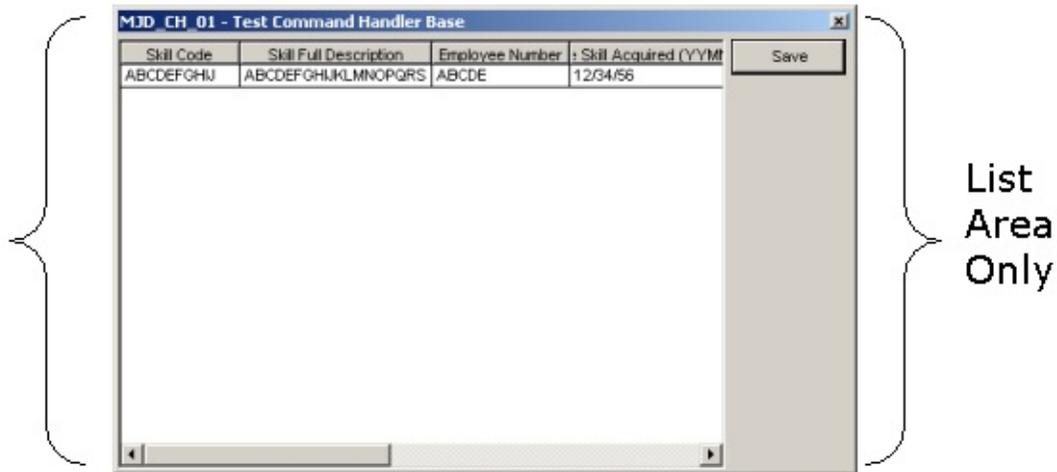
The "Top" of the command handler may contain single field instances arranged to flow down the viewing area. The "Bottom" of the command handler contains a list of fields arranged across the viewing area.

In other words, this is the classic header/details style layout. There are of course variations on this:

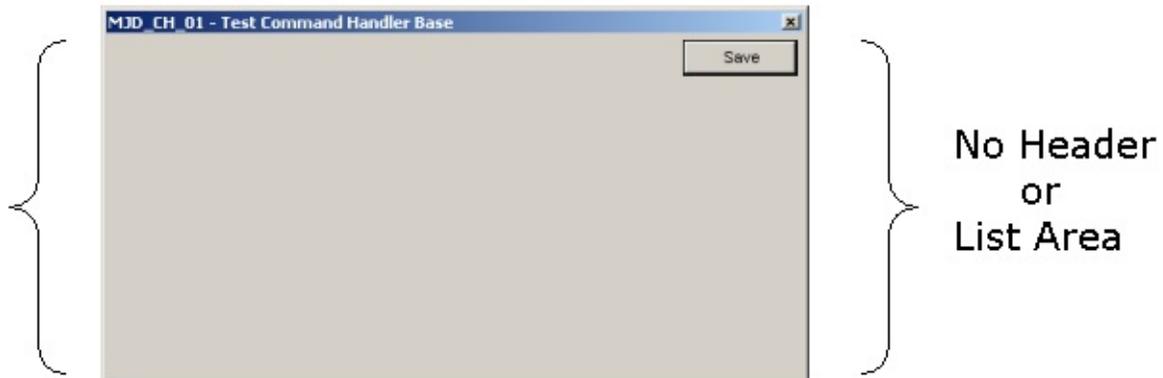
Header fields only:

The screenshot shows a window titled "MJD_CH_01 - Test Command Handler Base". The top section, labeled "Header Area Only", contains seven form fields: "Employee Number" (value: ABCDE), "Employee Surname" (value: ABCDEFGHIJKLMNOPQRST), "Employee Given Name(s)" (value: ABCDEFGHIJKLMNOPQRST), "Employee Salary" (value: 123,456,789.12), "Department Code" (value: ABCD), "Section Code" (value: AB), and "Start Date (DDMMYY)" (value: 12/34/56). A "Save" button is located to the right of the first three fields.

List fields only:



Neither Header nor List fields:



By examining the default file associated with this command handler and the header / details fields you have specified (or not) the code generator will attempt to anticipate (and generate) default code that is structured to fill in the form details.

The generated code is usually incomplete and it needs your input to be truly executable.

This is a [Program Coding Assistant](#) property.

File Prefix to be used for MS-Excel (Business object properties, Instance List tab)

This setting is only applicable if [Allow Instance List to be sent to MS-Excel](#) is checked. This contains the prefix that is used to construct the name of the Comma Separated Variable (CSV) file produced when the end-user sends the instance list to MS-Excel.

The file is created in the user's temporary directory, with a name of:

<<Prefix>>_<<Current date/time>>.csv

This property is in the Business Object [Instance List/Relations](#) tab.

Filter Tab Location

Use this option to specify where the tabs of the filter tab folder of this business object are displayed. The tabs can be displayed on the top, bottom, left or right of the tab folder.

Applies to **Windows** and **.NET** applications.

This property is in the [Filter Settings](#) tab.

Filter Tab Style

Use this option to specify how the filter tab folder for the selected business object will be displayed. Tabs is the normal tab sheet display, Buttons and Flat Buttons display the tabs as buttons.

FlatButtons should only be used with [Command Tab Location](#) Top.

Applies to **Windows** and **.NET** applications.

This property is in the [Filter Settings](#) tab.

First Time Only

Indicates that the introduction is only shown once during a session.

This property is in the Framework [Startup](#) tab.

Fixed / Default Values

Specify the default value(s) that are to be used for this custom property. For fixed lists two values are specified for each entry. One is the value that is shown to the Administrator, the other is the value that is accessed programmatically.

For example:

Value Caption to Display to User

CA California

NY New York

MN Minnesota

This property is in the [Custom Properties](#) tab.

Focused Input Field Style

This property allows Framework designers to specify the style of input fields on web applications when they get focus. In this way the field that is receiving input can be highlighted for the user.

The value specified here is passed as `cssText` to the `style` property of the HTML element representing the field on the Web page. As such it must be validly formatted as a style string that would be used in JavaScript or on an HTML page or unexpected results may occur. However if the style string, or some part of it, is not properly formed it will normally be ignored.

Valid examples of style strings, and therefore this property, are:

```
background-color:azure;border:4px double red  
background-color:#FDF5E6;border:3px ridge #C0C0C0  
color:rgb(255,0,0);border:3px inset rgb(200,200,200)
```

NOTE: To avoid this property replacing an input field's existing style use a prefix of 'APPEND=' for the property.

To make the previous examples add to existing styles use:

```
APPEND=background-color:azure;border:4px double red  
APPEND=background-color:#FDF5E6;border:3px ridge #C0C0C0  
APPEND=color:rgb(255,0,0);border:3px inset rgb(200,200,200)
```

Refer to the Microsoft JavaScript documentation for more details about valid input strings to the `Style` property.

This property is in the [Web/RAMP Details](#) tab.

Framework Fatal Error

This option only applies to **Windows** Framework applications.

If you select this option, a Framework fatal error will be initiated when a user has exceeded the number of allowed sign on attempts.

How Framework fatal errors are handled will depend on how the Framework has been set up, but usually the Framework will send an email to the administrator and then shut down. Any subsequent attempts to sign on as the same user profile after restarting the Framework will result in the same action.

To re-enable this user profile, the administrator must deselect the User Is Disabled checkbox for this user in the User Details tab.

This property is in the Framework [User Administration Settings](#) tab.

Function Handling Table Data storage

This is the function that will store any table data entered in the Data tab.

Tables where the data is stored in the shipped generic table data file (FPTAB) can share the default table data handler UF_SYSBR/UFU0010, or use their own version of it.

Tables where the data is stored in a user defined physical file can create their own table data handler function. See the data handler function UF_SYSBR/UFU0011 for an example. This function reads and writes data from/to file DEPTAB, for the table DEPTAB.

Tables where the data comes from hard coding in the table data handler function can create their own table data handler function. See the data handler UF_SYSBR/UFU0012 for an example of this type of data handler. It supplies the data for the SEX Table.

Tables can also get their data from a flat file, using a table data handler function similar to UF_SYSBR/UFU0013. They could conceivably also store saved data on a flat file.

This property is in the Code Table [Definition](#) tab.

Generate Web Pages

Use this option to enable and disable the production of start up web pages while doing development.

If you have 10 languages defined then when you save your Framework pages will be generated for all 10 languages and then copied to your HTTP server.

This can be a time consuming process, so you can selectively disable language(s) while you are developing and testing to reduce this time. When you have completed your development cycle you can then (re)enable them to produce final copies of all the HTML start up pages ready for deployment.

This property is in the [Web/RAMP Details](#) tab.

Groups this user belongs to

Framework authority can be specified for groups of users.

Check the groups whose authority you want to give to this user. When you edit this user's authority, the authority that this user receives from any groups they belong to will be shown as shaded and non-input capable.

Authorities are ADDITIVE, so:

- if the individual user is not authorized to the object, but one of the groups they belong to is authorized, the user will be able to use the object.
- if the individual user is authorized to the object, and the user is attached to a group that is not authorized to the object, the user will still be able to use the object
- if the user belongs to two groups, and one group is authorized to an object and one group isn't, the user will be able to use the object.

To create a group, press the New Group button. The group authorities are edited in the same way as an individual user's authorities are.

This property is in the [User Details](#) tab.

Height

Enter the height of the image/web page in pixels, to be displayed on the Sign On form.

This property is in [User Administration Settings](#) tab.

Help Text

Specifies any help text to be associated with the custom property and shown to the administrator to aid him/her in selecting the correct property value.

This property is in the [Custom Properties](#) tab.

Hide All Other Command Tabs

Select this option to indicate:

- When this command is displayed any other command tabs at the same level are to be hidden and no tabs at all should be displayed.
- This command is not to appear as a tab when the other command tab sheets are displayed.

For example, a business object "Customer" might have these instance level commands:

- Details (the default command)
- Notes
- History
- Delete

When you click on a customer in the instance list, a tab folder with tabs Details, Notes, History and Delete will appear.

The Details tab will be at the front and the Details command handler will be executed to fill in the appropriate details.

If the Hide All Other Command Tabs option is selected for the Delete command, only the tabs Details, Notes, History will be displayed when you click on a customer in the instance list.

If you click on a customer in the instance list and then execute the Delete command from a right mouse popup menu, the toolbar or the menu bar, the Delete command handler will be displayed with no tabs at all visible.

This property is in the [Commands Enabled](#) tab.

Hint

You can optionally enter a description for the object to be displayed in a hint.

This property is in the [Identification](#) tab.

Host Name or IP Address

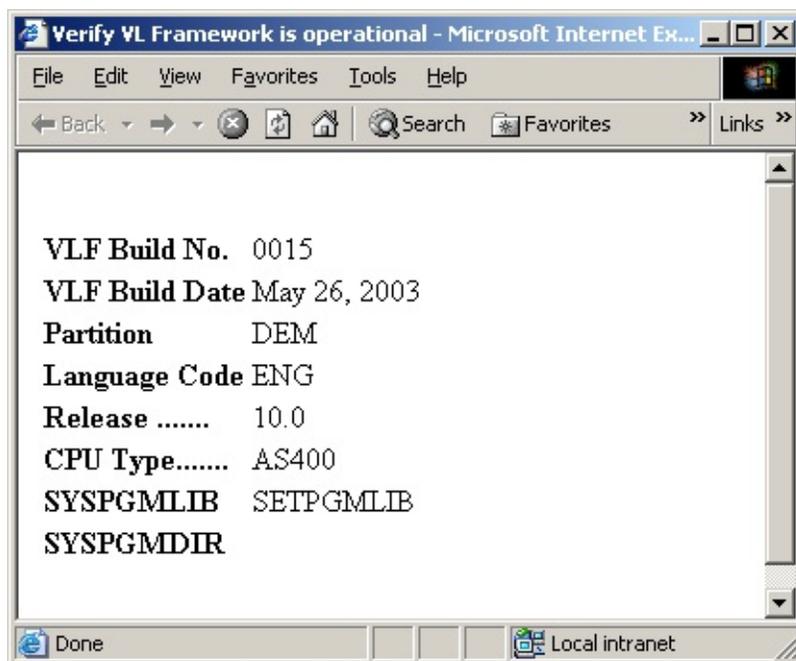
Specify either the IP address in nnn.nnn.nnn:pp format or the host name of your LANSA for the Web development system server.

After changing this value you should always verify the address by clicking the Verify Web button.

The resulting display should something like the following sample (note that the background colors, header/footer areas and build numbers and dates may be different to this sample).

Check that the partition, language and LANSA system program library values are what you expect to be returned from your development server.

If the verification process fails you should fully resolve the problem before proceeding further.



This property is in the [Developer Preferences – Web Server](#) tab.

IBM i Host Server Mapper Name / IP address

Name or IP address of the IBM i Server Mapper. Supports full 40 character long IPV6 type addresses.

This property is in the [Server Details](#) tab.

IBM i Host Server Mapper Port

Port of the IBM i Server Mapper to connect to. Defaults to IBM default.

This property is in the [Server Details](#) tab.

Icon

Choose an icon from the list. The icon will be assigned to the currently selected object.

Note that you can right-click the list of icons to bring up a pop-up menu to toggle between large and small icons.

You may want to review [How do I enrol my own bitmaps and icons?](#).

This property is in the Framework [Icons](#) tab and the [Bitmaps and Icons](#) tab

Icon and Bitmap Enroller

Specify here the name of your icon and bitmap enroller. Refer to the shipped source code of component UF_IB001 for details of how and when you should change this property.

This property is in the [Framework Details](#) tab.

IIP - User Signon Function Name

This is the name of the LANSAR DML function that will be called on the web server to validate the user profile and password of the user attempting to access browser based Framework.

The default function is named UFU0001.

You must have a validly executable user signon function in all Web browser applications.

Refer to [Imbedded Interface Points \(IIPs\)](#) for more details.

This property is in the [Web/RAMP Details](#) tab.

IIP – Function to return web user authorities

In Windows applications the Framework authority model may be replaced by user code in method avCheckAuth. See the shipped example in Visual LANSA component UF_SYSTM (method avCheckAuth) for more details.

In web browser applications you may also replace the Framework authority model. To do this you need to write a LANSA function. The name of the function is specified in this field. A shipped example of this type of function can be found in function UFU0016 in process UF_SYSBR.

This property is in the [Web/RAMP Details](#) tab.

Import Users Imbedded Interface Point

This is the name of a reusable part that will be executed when the Administrator presses the Import Users button in the User Details tab (accessed using the Users option of the Administration menu).

The specified reusable part collects user data from external sources and creates an XML file of User Data from it. The Framework then automatically imports the User Data into the Framework and creates users from it.

See the source of shipped example component UF_IMPUS for more details on how to create a component like this.

Also see section [Export or Import of Full or Partial User Data](#).

This property is in the Framework [User Administration Settings](#) tab.

Image File

The name of the image file to be displayed at startup, without path information.

Place the file in the Execute directory of your partition (for example C:\X_WIN95\X_LANSA\X_DEM\EXECUTE).

Typically image files need to be deployed with your application.

This property is in the Framework [Startup](#) tab.

Image / Web Page to Display on Form

The name of the image file to be displayed on the Sign On form, without path information.

Place the file in the Execute directory of your partition (for example C:\X_WIN95\X_LANSA\X_DEM\EXECUTE).

Typically image files need to be deployed with your application.

Alternatively this may be specified as a URL. In this case the web page will be displayed instead of an image.

The property value will be recognised as a url if it starts with http: or https: or file: or *AUTO*.

AUTO implies that the html file is in the partition execute directory, and will be displayed using the web browser.

Examples

Web pages:

<http://www.lansa.com>

<https://www.lansa.com>

A html file that exists locally on the PC:

file: C:\Program

Files\LANSA\X_WIN95\X_LANSA\x_dem\execute\ENG_VF_UM087_01.htm

or

*AUTO*ENG_VF_UM087_01.htm

This property is in the Framework [User Administration Settings](#) tab.

Images Folder

For a System i web server this is the folder where you have put LANSAs for the Web images (the LANSAs for the Web Images folder). This is the actual IFS directory, not the HTTP alias assigned to it.

For pre Version 11.0 LANSAs systems the **default** LANSAs for the Web images folder is LANSAIMG.

For brand new Version 11.0 LANSAs systems, the **default** LANSAs for the Web images folder is /LANSA_dc@pgmlib/webserver/images

Type in the IFS directory name of your LANSAs for the Web images folder.

For a Windows web server this is the HTTP alias name of the folder where you have put your LANSAs for the Web images. If your Windows server is using IIS, by default the HTTP alias name assigned to it is Images. If you are unsure, please contact your Web Server Administrator.

Type in the Windows web server images alias name.

Verify the path name is correct and that the folder is accessible by clicking on the Verify button.

Note that by default, Directory Browsing is disabled in IIS. To enable Directory Browsing, go to the Control Panel/Administrative Tools and double click on Internet Information Services. Expand the local computer, then the Web Sites folder, then the Default Web Site folder. Right-click on the Images folder, select Properties and tick the Directory Browsing box.

The resulting Internet Explorer display should show the index of the images folder.

If the verification process fails, stop and resolve the problem before proceeding further.

This property is in the [Developer Preferences – Web Server](#) tab.

Import Users from XML

Use this option to import user data from an XML file.

See [How to Import User data from an XML file](#).

This property is in the [Authorities](#) tab.

Inactive Table Entry indicator

In some circumstances it is useful to be able to distinguish between table entries that are live, and inactive table entries that are present only because they belong to historical data. The inactive table entries need to be present so that historical data is displayed correctly, but they shouldn't be used when creating new data.

To flag a table entry as inactive, create a one character field field to hold the inactive indicator and then choose that field in the *Inactive Table Entry* indicator drop down list.

This property is in the Code Table [Definition](#) tab.

Include Default Save Button and Logic

Check this option if you want your generated code to include a "Save" button and an associated Click event handling routine.

This is a [Program Coding Assistant](#) property.

Include Default Search Button

Check this option if you want the generated code for this filter to include a default search button and an associated Click event handling routine.

This is a [Program Coding Assistant](#) property.

Include Layout Managed Button and Field Areas

Check this option if you want the generated code for this filter to include a layout managed field and button areas.

This is a [Program Coding Assistant](#) property.

Include uInitialize Routine in Command Handler

Check this option when you want the generated code for this command handler to include an initialization routine (named uInitialize). Typically only command handlers that [Stay Active](#) use initialization routines.

This is a [Program Coding Assistant](#) property.

Include uInitialize Routine in Filter

Check this option when you want the generated code for this filter to include an initialization routine named uInitialize. Typically only filters that [Stay Active](#) use initialization routines.

This is a [Program Coding Assistant](#) property.

Include uTerminate Routine in Command Handler

Check this option when you want the generated code for this command handler to include a termination routine (named uTerminate). Typically only command handlers that [Stay Active](#) use termination routines.

This is a [Program Coding Assistant](#) property.

Include uTerminate Routine in Filter

Check this option when you want the generated code for this filter to include a termination routine named uTerminate. Typically only filters that [Stay Active](#) use termination routines.

This is a [Program Coding Assistant](#) property.

Input Method

Specifies the way that the value(s) for the custom property should be input. The options available are:

- Single A single value should be input. May be selected for types Alphanumeric, Numeric or Boolean.
- List A list of values should be input. May be selected for types Alphanumeric and Numeric.
- Fixed List The administrator should choose from a fixed list of options. May be selected for types Alphanumeric, Numeric or Boolean.

This property is in the [Custom Properties](#) tab.

Instance Command

Select this option if you want the selected command to apply to the instances of the business object. Instance commands are displayed when you right-click an instance in the [Instance List](#).

Whether a command is best implemented as applicable to the business object itself or its instances depends on the context.

For example, let's assume you have a business object Employee with instances such as 'employee A1234 Veronica Brown'. To allow the user to enter new employees, you would create the command New. Logically this command applies to the business object itself, Employee. Therefore, you do not select the Instance Command option for command New.

To view details of an individual employee (in other words an instance of the Employee business object) you could add a command Details. You need to make this command an instance command because it will display details of the individual employees (such as phone number, salary etc.), not about the business object. So for Details you would select the Instance Command option.

An instance command handler can be written in such a way that it processes several instances at once. To see how this works see the Programming Techniques application -> Basic -> Selected, Current or All Entries.

This property is in the [Commands Enabled](#) tab.

Instance Command Presentation

Use the Instance Command Presentation option to control how the commands for the business object instances are displayed:

- | | |
|-----------------------------|--|
| Use all of the window | The command handlers take up all of the right-hand side of the window. |
| Use part of the window | The command handlers take up the bottom part of the right-hand side of the window. |
| Separate normal window | The command handlers are shown in a separate window. |
| Separate stay on top window | The command handlers are shown in a separate window that always stays on top of the main Framework window. |

This property is in the [Command Display](#) tab.

Instance List Relationship Summary

This is a list of the instance list relationships between all the business objects in the Framework.

The list shows the [Caption](#) and the [User Object Name/Type](#) of the business object and any child or peer-to-peer relationships it has with other business objects.

For more information, refer to [Instance Lists with Different Types of Objects](#).

This property is in the Framework [Instance List Relationships Summary](#) tab.

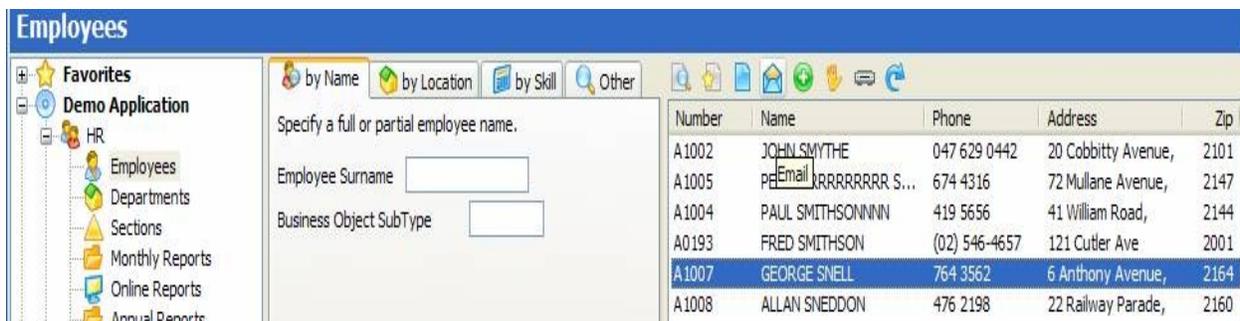
Instance List Tool Bar Location

Applies to **Windows** and **.NET** applications.

Set this option to Top, Left, Right or Bottom to cause a tool bar of the commands associated with objects in the instance list to be displayed in the specified location. Set this option to None to prevent an instance list tool bar from being displayed.

The existence and location of any instance list tool bar is an application designer controlled feature.

This option only applies to the shipped instance browser.



This property is in the Business Object [Instance List/Relations](#) tab.

Instance List Tool Bar Height or Width

Where an instance list tool bar is displayed use this option to control the height (for Top or Bottom location) or width (for Left or Right location) of the tool bar and the images that appear on it.

This property is in the Business Object [Instance List/Relations](#) tab.

Instance List Tool Bar Text Location

VLF.WIN only.

Specifies if and where the text associated with a command on an instance list tool bar is to be displayed. Options are <none>, Left, Right, Top and Bottom.

For example these settings:

Instance List Tool Bar Location	Top
Instance List Tool Bar Text Location	Bottom
Instance List Tool Bar Height or Width	55

Produce an instance list tool bar like this example:

 Details	 Documents	 Events	 Images	 Notes	 Reports	 TimeSheets
Name	Address Line 1	Code / Id	Business Phone	Address Line 2		
BROWN,VERONICA	12 Railway Street	A0070	(02) 9647 2788	Baulkham Hills		
BLOGGS,FRED JOHN ALAN	70 MAIN STREET	A0090	654 6475 X432	NEWTOWN NSW		

These settings:

Instance List Tool Bar Location	Top
Instance List Tool Bar Text Location	<None>
Instance List Tool Bar Height or Width	22

Produce an instance list toolbar like this example:

      						
Name	Address Line 1	Code / Id	Business Phone	Address Line 2	Home Phone	
BLOGGS,FRED JOHN ALA	70 MAIN STREET	A0090	654 6475 X432	NEWTOWN NSW	344-2234454545	Reports
SMITHSON,FRED	121 Cutler Ave	A0193	(02) 354-5647	Windsor	(02) 546-4657	

This property is in the Business Object [Instance List/Relations](#) tab.

Internal Identifier

This is an internal identifier automatically generated by LANSAs to uniquely identify this object. The identifier is alphanumeric.

This property is in the [Identification](#) tab.

IP Address

Specify the IP address of the host where RAMP-TS has been installed.

Note to VLF-WEB users:

- The RAMP-TS session must be in the same domain as the VLF-WEB host.
- The RAMP-TS server's IP address must be included in the list of trusted sites in your browser.

This property is in the [Server Details](#) tab.

IP Address and Port Number

For RAMP-NL users only.

The Internet Address and Port number of the Host to connect to. These properties are ignored if a Server Name has been specified.

Note that if you specify an IP Address and Port Number, RAMP-NL will entirely bypass the newlook connection definition.

This means that RAMP-NL will derive the licensing username and password from the Framework logon profile, not the newlook connection definition. Consequently the newlook licensing signon dialog will never be displayed.

Similarly, RAMP-NL will ignore all other options specified in the newlook connection definition, such as screen size and date format. Therefore, if you need to for example support 27 x 132 screens, you cannot use the IP Address and Port Number option.

We recommend you only use this option if you are fully aware of the potential consequences.

This property is in the [Server Details](#) tab.

Intro Caption

Specifies the caption associated with a URL (e.g.: "Visit our Web Site for more information")

This property is in the Framework [Startup](#) tab.

Intro URL

Specifies how the URL of the web page is handled when it is displayed. The allowable values are:

Browser	The display of the URL is displayed in a separate browser window. This is the default.
Current Window	The display of the URL is imbedded inside the Framework's introductory panel.

This property is in the Framework [Startup](#) tab.

Keep newlook SID File Versions

Select this option to keep backup copies of the newlook SID files before making changes using Dynamic Naming. For more information refer to the RAMP Guide.

Note that newlook licensing features may limit the use of this option in some RAMP environments. Please contact your product vendor for further information.

This property is in the [Framework Details](#) tab.

Keep Versions in Subfolders

If you select this option and the [Keep XML File Versions](#) option is checked, old versions of the XML will be stored in a subfolder of the usual storage location, called\VF_Versions_\

Usually this is the <<partition execute directory>>\VF_Versions_\

This can help to keep your partition execute directory tidy.

Note that if you change this option:

- Any existing XML file versions will not be moved to the new folder - you should do this manually.
- Any version folders no longer required will not be deleted. You should do this manually after moving any XML files versions in the folder as appropriate.

This property is in the [Framework Details](#) tab.

Keep XML File Versions

Select this option to keep back up copies of the XML files. In this way you can keep copies of your Framework definition, user definitions and server definitions.

If you do not keep copies of your Framework you will not be able to revert to previously saved versions in the event of an accidental object deletion. See [I have just deleted an object and want to get it back again?](#)

This property is in the [Framework Details](#) tab.

Keys of the View

Map the keys in the instace list with the keys in file or view that is used to read the child object data.

This is a [Program Coding Assistant](#) property.

Keys from the Instance List

Choose the keys in the instance list which are used to map to the child data.

This is a [Program Coding Assistant](#) property.

Language Field

This is used for multilingual table data (see the shipped Title table for an example of a multilingual table).

If multilingual table data is not required, set this to the value "No Language Field - Monolingual Table Data"

A table does not necessarily have to be multilingual just because the partition is multilingual. But if multilingual data is required, use this parameter to specify which table field will contain the language code for a table row. A language field must also be a key field.

This property is in the Code Table [Definition](#) tab.

Languages

This list shows details of all the languages for which web browser capable versions of the Framework will be produced.

This list should always include the language you are currently executing the Framework in. It may include other languages valid in the partition in which you are working. It should NOT include languages that are not valid in the partition in which you are working.

This property is in the [Web/RAMP Details](#) tab.

LANSA Language Code

This is the LANSAs language code associated with the language you are viewing or wish to define. This language code must specify a language that is validly defined to LANSAs in the partition in which you are currently working.

If you are working in a mono-lingual partition you should define a single language with code NAT and caption National Language.

When you type in a new language code the Framework will default the Caption, Meta Tag, CCSID and DBCS options for you. Unless you know exactly what these options do you should use the default values.

Typically LANSAs systems use these language codes:

Code Language

CES	Czech
DEU	German
DUT	Dutch
ENG	English
FIN	Finnish
FRA	French
GRE	Greek
HEB	Hebrew
ITL	Italian
JPN	Japanese
KOR	Korean
NOR	Norwegian
POR	Portuguese
SCHI	Simplified Chinese
SWE	Swedish
SFRA	Swiss French

TCHI Traditional Chinese

TUR Turkish

This property is in the [Web/RAMP Details](#) tab.

Last Changed

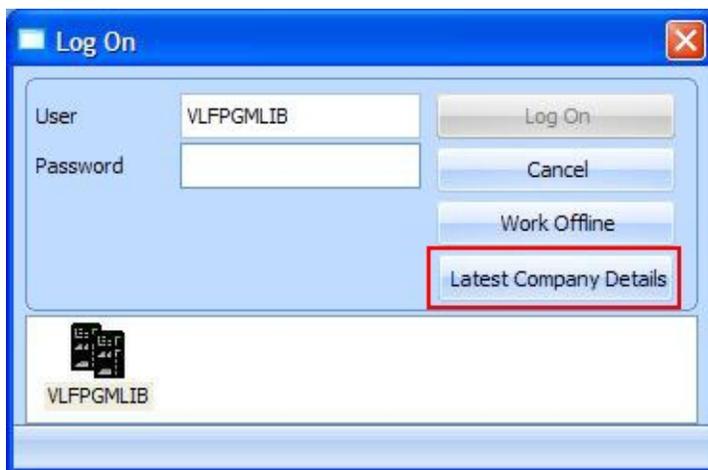
This field indicates when the object was last changed and the user who made the change.

This property is in the [Identification](#) tab.

Launch Button Caption

If this property is specified (i.e. it is not blank), a button will appear on the user's signon screen. When the button is clicked, a window will be opened with the URL specified.

This can be used to launch any URL from the user's sign on screen. For example it could be used to launch a website showing the latest company details in VLF.WIN:



VLF.NET:



And VLF.WEB:



This property is in the Framework [User Administration Settings](#) tab.

Launch from Status Bar

Select this option to display the Launch button. The button enables users to launch applications directly from the status bar.

The launch button is available only when the Framework is executed using RenderType M.

If you Allow Users to Switch Views, the Launch button is always visible .

If you do not Allow Users to Switch Views and the Launch from Status Bar option is selected, the application/application view buttons are shown:



See [Launching Applications from the Status Bar](#).

This property is in the [Framework Details](#) tab.

Launch URL (Windows)

The URL that will be used to open a window when the launch button on the Windows sign on screen is clicked.

(The launch button must have a caption to make it appear.)

The URL may start with:

- | http://
- | https://
- | *AUTO*

AUTO will look for a html file in the partition execute directory, for example:

```
*AUTO*df_demo_hr_page1.htm
```

A typical URL to launch a wam webroutine is:

```
http://<<The Host>>:<<port>>/CGI-BIN/lansaweb?webapp=<<my wam  
name>>+webrtn=<<my  
webroutine>>+ml=LANSA:XHTML+part=DEM+lang=ENG
```

This property is in the Framework [User Administration Settings](#) tab.

Launch URL (Web / .Net)

The URL that will be used to open a window when the launch button on the web or .Net sign on screen is clicked.

(The launch button must have a caption to make it appear.)

The URL may start with:

- | http://
- | https://
- | *AUTO*

AUTO will look for a html file in the images directory, for example:

```
*AUTO*df_demo_hr_page1.htm
```

A typical URL to launch a wam webroutine is:

```
http://<<The Host>>:<<port>>/CGI-BIN/lansaweb?webapp=<<my wam  
name>>+webrtn=<<my  
webroutine>>+ml=LANSA:XHTML+part=DEM+lang=ENG
```

This property is in the Framework [User Administration Settings](#) tab.

List Panel

Drag here the fields you want to be displayed in the list of child objects.

This is a [Program Coding Assistant](#) property.

Load Path

For RAMP users only.

Enter the path where the RAMP runtime files are to be located. These files are: VF_SY120.htm and VF_SY120.js.

The load path defaults to the current LANSAs system's partition execute directory. Usually there is no need to change the default.

This property is in the [Server Details](#) tab.

Location for Buttons

Note: WEBEVENT command handlers and filters are a deprecated feature. Do not use them for new work.

Use this option to specify the location of the buttons.

This option only applies to WEBEVENT command handlers and filters.

This property is in the [Commands Enabled](#) tab the [Filter Snap-in Settings](#) tab.

Log off Inactivity Timeout

Specify here the period of inactivity, in minutes, after which the user is automatically logged off.

After certain periods of inactivity it may be assumed that the user no longer requires the application. Automatically closing the application will free system resources and improve application security.

Activity is defined as selection of an application or business object or the execution of a command.

A setting of zero will disable this feature.

Note that when the VLF is running on touch device, this option is disabled. You should use device specific timeout settings to control access to the device.

Note that you can use either a Log on or a Log off Inactivity Timeout, but you should not use both.

This property is in the Framework [User Administration Settings](#) tab and the [User Details](#) tab.

Log on Inactivity Timeout

Specify here the period of inactivity, in minutes, after which the user is asked to confirm their log on password.

Long periods of inactivity may present a security risk for an active application. Asking for confirmation of the Log on password will ensure only authorised users have access to the application.

Activity is defined as selection of an application or business object or the execution of a command.

If Framework users are not specified or if this property is set to zero this feature is disabled.

Note that when the VLF is running on touch device, this option is disabled. You should use device specific timeout settings to control access to the device.

Note that you can use either a Log on or a Log off Inactivity Timeout, but you should not use both.

This property is in the Framework [User Administration Settings](#) tab and the [User Details](#) tab.

Major Comment Separator

Specify the character that should be repeated to delimit major comment blocks in your generated code. Typical choices include:

Equal: =====

Minus: -----

Plus: ++++++

Underscore: _____

This is a [Program Coding Assistant](#) property.

Marker for Code Requiring Manual Completion

Specify the string to be inserted in generated code where manual coding is required to complete the code. Typical choices include:

```
=> TO BE SPECIFIED <=
```

```
????????????????
```

Avoid using values starting with << for this setting. Avoid using the same character here as you use for the field prefix (e.g.: #).

This is a [Program Coding Assistant](#) property.

Maximum Decimals

For numeric custom properties a maximum number of decimals in the range 0 to 5 may be specified.

This property is in the [Custom Properties](#) tab.

Maximum Entries in List

When the input method is List or Fixed List specify the maximum number of entries that can exist in the list.

This property is in the [Custom Properties](#) tab.

Maximum Length

For alphanumeric custom properties values a maximum length in the range 1 to 256 may be specified.

This property is in the [Custom Properties](#) tab.

Maximum Signon Attempts Allowed

Enter in this field the number of unsuccessful signon attempts that should be allowed before a user is rejected.

In web applications a rejected user is also disabled and must be specifically re-enabled by an administrator.

In Windows applications rejected users are forced to exit the Framework with an error message, but they are not disabled.

This property is in the Framework [User Administration Settings](#) tab.

Maximum Web Password Length

Change this property to enable the maximum allowable length for passwords to be extended to 32 characters for Web applications. Minimum value allowed is 10, maximum value allowed is 32.

This property is in the [Web/RAMP Details](#) tab.

Meta Tag

Specifies the <META> tag that will be used in HTML pages produced by LANSAs for the Web when using this language. Normally this value is defaulted for you and you should use the default value unless instructed otherwise by your product vendor.

This property is in the [Web/RAMP Details](#) tab.

Minor Comment Separator

Specify the character that should be repeated to delimit major comment blocks in your generated code. Typical choices include:

Equal: =====

Minus: -----

Plus: ++++++

Underscore: _____

Blanks:

Avoid using the same character here as you use for the field prefix (e.g.: #).

This is a [Program Coding Assistant](#) property.

Mock Up RAD-PAD

The RAD-PAD mock up filter or command handler creates a tab sheet where you can simply type in a description and add pictures from the [Command or Filter Handler](#). The mock up tab sheet contains no code and has no functionality.

A real filter limits the items shown in the instance list. A real command handler controls what is displayed and what happens when the user selects a command.

When you are finished with prototyping your application, specify the name of a [Command or Filter Handler](#) to replace the mock up tab sheets.

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

MTXT String Loader

Specify here the name of your end-user visible multilingual string program loader.

The string loader exposes those *MTXT variables that are compiled into the Framework during the build. It allows you to use your own multilingual text for Framework captions such as the "Clear List" on the Instance list.

Refer to the shipped source code of RDML function UFU0003 in process UF_SYSBR for details of how and when you should change this property.

This property is in the [Framework Details](#) tab.

Multiline Tab Sheet Captions

Select this option if you want the tab sheet captions to be displayed on more than one line.

This option is only applicable when the tab style is Tabs and the tab location is Top or Bottom.

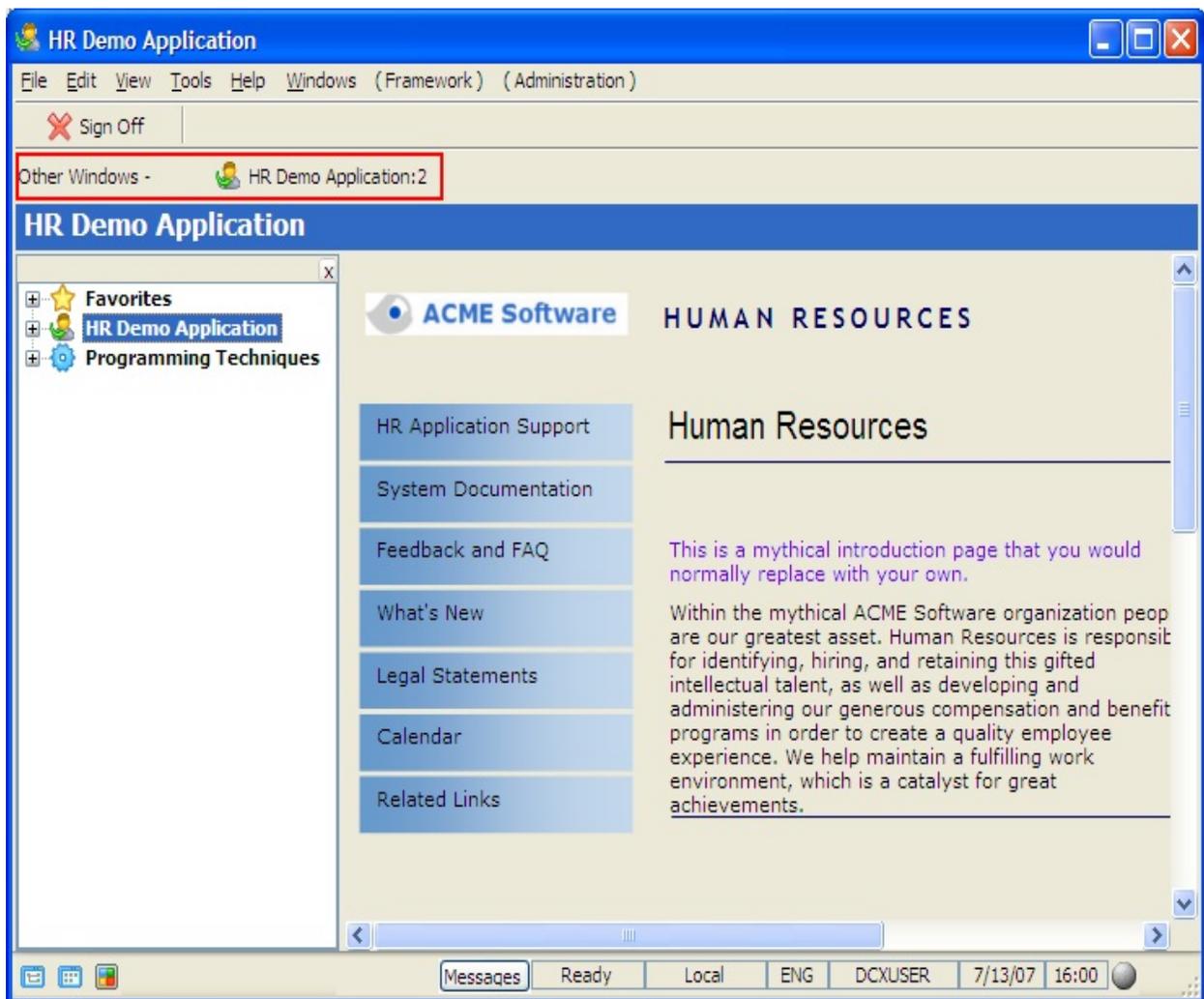
This property is in the [Command Display](#) tab and the [Filter Settings](#) tab.

Multiple Window Control Bar Location

Use this option to change the location of the Windows Control Bar that is displayed on Framework forms when more than one Framework window is open. This option can also be used to hide the Windows Control Bar.

The default location is Above Title Bar.

To hide the Windows Control Bar, select option None - Do not display control bar.



This property is in the [Identification](#) tab.

Name

Specifies the symbolic name to be associated with the custom property. This name is used to programmatically identify a property. They are only ever used by developers and are never seen by administrators. Names are at most 20 characters long and can only contain letters from the English alphabet (A -> Z), underscore (_) or 0 through 9.

This property is in the [Custom Properties](#) tab.

Name of User Set to be Used

When you have elected to store user and authority details in DBMS tables VFPPF06/07 (see [Store Users in XML File and Store users in DBMS Tables VFPPF06/07](#)) you may also specify a User Set.

User Sets may be used to

- Divide user details between different Frameworks.
- Share user details between different Frameworks.

For example, imagine you have two separate Frameworks named GENLEDGER and CRM defined.

Both Frameworks use the option to store user details in DBMS tables on the server.

Assuming that both Frameworks operate in the same LANSAPARTITION then they will both be use the same DBMS tables to store their user details.

Are the user profiles stored in the DBMS tables to be shared or to be separated?

To share the user details give both Frameworks the same User Set Name.

To separate the user details give both Frameworks different User Set Names.

The default User Set name is SYSTEM.

It is strongly recommended that you restrict User Set names to at most 10 contiguous uppercase characters of the English alphabet (i.e. A-> Z only with no imbedded blanks). This will minimize the risk of any code page conversion issues now or in the future.

Please note that separate really means separate.

User1 in user set A has absolutely nothing to do with User1 in user set B.

This property is in the Framework [User Administration Settings](#) tab.

.NET Component Class Name and Assembly

The component class name of the .NET component that is snapped in to the framework as a Filter, Command Handler or Relationship Handler and the name of the Assembly file that contains the class that is snapped into the Framework.

These values cannot be edited but must be set using the Find button.

See the VLF .NET Snap-in Components Guide for more information.

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

.NET Target Platform

The .NET Target Platform option specifies the target platform for the generated VLF.NET application.

The default value is AUTO which means that the runtime platform is determined by the operating system architecture. Setting the option to X86 will force the generated VLF-NET application to run in 32-bit.

Generally designers do not need to change this value unless their VLF.NET application has to specifically run as a 32-bit application.

An example of such a situation is when an application uses a 32-bit ActiveX control (such as Office Excel 2003 Web Component), it is necessary to force the generated VLF.NET application to run in 32-bit because 32-bit ActiveX cannot be hosted in 64-bit applications. In this case you would set .NET Target Platform to X86.

This property is in the [Framework Details](#) tab.

Nodes XML File

Use the Nodes XML File property to specify the name of the file where RAMP screen definitions for the Framework are stored. This file is located in the Execute directory of the current partition.

The default value is Vf_Sy001_Nodes.xml.

The file name before the .xml extension should not be longer than 14 characters if it is to be deployed using the Deployment Tool.

This property is in the [Framework Details](#) tab.

Number of Additional Windows a User can have Open Concurrently

Use this option to regulate the number of additional windows the user can have open at the same time. Maximum value is 256.

Opening multiple windows of course uses more workstation resources, so you should avoid allowing an excessive number of windows to be concurrently open.

Also see [Windows Resource Usage](#).

This property is in the [Identification](#) tab.

Object Command Presentation

Use the Object Command Presentation option to control how the commands for this object are displayed:

Automatic	All the window is used except when the command is displayed for business object that has one or more filters.
Use all of the window	The command handlers take up all of the right-hand side of the window.
Use part of the window	The command handlers take up the bottom part of the right-hand side of the window
Separate normal window	The command handlers are shown in a separate window.
Separate stay on top window	The command handlers are shown in a separate window that always stays on top of the main Framework window.

This property is in the [Command Display](#) tab.

Open Latest Demonstration System

Select this option to open the Latest Demonstration System (VF_SY001_System_LastShipped.XML).

This property is in the Select Framework dialog.

Optional Arguments

Enter optional numeric or alphanumeric values that are passed into the command or filter handler. Using optional arguments allows a single handler to be easily reused for different tasks.

For example, **DF_WEBBR** is a demonstration command handler that displays web pages. By specifying **DF_WEBBR** as the command handler and putting the required URL (e.g.: www.lansa.com) into Argument 1 will cause that web page to be displayed.

The optional arguments have no meaning with mock-up handlers because they have no functionality.

This property is in the [Commands Enabled](#) tab.

Optional Mapped Drives - Images Folder and Private Working Folder

If you are building browser applications, then, the Visual LANSA Framework will need to copy data from your Visual LANSA workstation to your LANSA for the Web images folder and to your Private Working Folder (these folders reside on your LANSA for the Web server system).

It can do this by using FTP commands or DOS COPY commands.

DOS COPY commands are usually faster.

To use DOS COPY commands you need to have a mapped drive that allows access to both folders.

The mapped drive can be in either \\server\xxx\xxxx format or in <drive>:\xxxx\xxxx format (often drive I: is used for such mapped drive assignments to System i servers).

If you have mapped drive network access to your Images and Private Working folders specify both paths in full.

You should use the Verify Button for both paths. A message box should be displayed indicating the success of the verification process. If the verification process fails you should resolve the problem before proceeding further.

See [Images Folder](#) for more details about the LANSA for the Web Images folder.

See [Private Working Folder](#) for more details about setting up your Private Working folders.

Also see [Script for Uploading to your LANSA for the Web Folders](#).

This property is in the [Developer Preferences – Web Server](#) tab.

Overall Theme

Applies to **Windows** Framework applications. Similar options can also be used with Framework **.NET** applications (see [Web Application Start Options](#)).

Visual Themes are an easy-to-activate substitute for visual styles. They produce a dramatic improvement in appearance with very little effort.

To use visual themes you must:

- Be executing in a Visual LANSAs 11.5 or later environment
- Be at Visual LANSAs Framework level EPC831 or later.
- Select a theme using this property

Themes are intended for use on screens with 32 bit color

When a theme is selected the appearance of the entire Framework changes to the new theme immediately, including your own command handlers and filters.

If you want to use a visual theme, but don't want it to change the appearance of a filter or command handler, you can deactivate themes for a panel or the whole component by changing the ThemeStyle property to None.

When using a theme, the Visual LANSAs Framework switches on the ThemedReadOnly option. This means that your read_only fields will be displayed with an appropriate read_only background for the theme, and it will override any visual style you have applied to the field. And it means that in future, you do not need any special coding to make read-only fields look different to input capable fields.

Note that if the option End User Can Change Theme is selected, any choice by the end-user using the Overall Theme option in the Windows menu overrides this value.

This property is in the [Visual Styles](#) tab.

Own Window Size

For commands that will appear in their own command window (see [Object Command Presentation](#)) you can specify the initial height and width that should be used for the command window.

Separate values for Windows and Web Browser contexts may be specified.

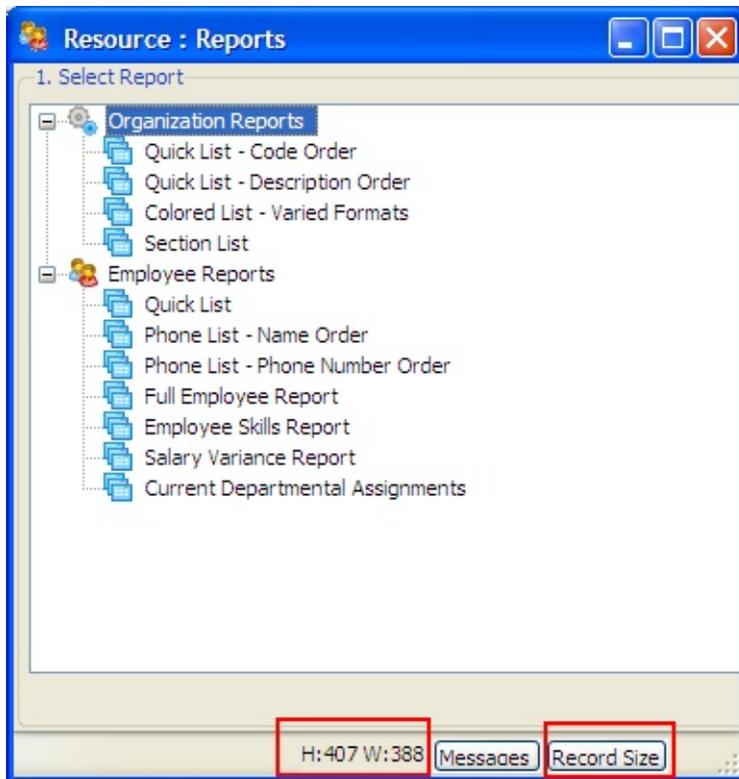
All values specified are in pixels.

If you do not specify sizing values (or set them to zero) the Framework will attempt to guess an appropriate value for you.

Any sizing values that you specify only apply the first time the command window is presented for the specific command involved. Subsequent redisplay of the command window within a signed on session will reuse the size that the command window had when it was last presented (i.e. the Framework will remember the window size for a each specific command within a signed on session).

To assist you in sizing windows in Windows applications you may execute the command window and then size the window to the exact size you require. If you are working as an application designer, the current window size and a button Record Size are available to help you.

For example:



This command window shows the current window size to be height 427 and width 563.

Click the Record Size button to update the size details into the definition of the command currently being executed, which in this example is the "New" command within the "Employees" business object.

In Web browser applications one technique that can be used to determine the size a window is:

- Start MS-Paint and create a small image (eg: 10 x 10).
- Execute the browser Framework application and display the window in question. Size to the exact size required.
- Use Alt-Print Screen to copy the window image to the clipboard.
- Paste the window image into MS-Paint. Allow MS-Paint to expand the current image.
- In MS-Paint check the pasted image attributes. This will display the exact width and height of the pasted window image.

This property is in the [Commands Enabled](#) tab.

Partition

Enter here the partition to connect to on the server.

This property is in the [Server Details](#) tab.

Partition is Enabled for RDMLX

If the partition on the server is RDMLX enabled and you are going to use RDMLX server side components, functions or files within your Framework application, you should select this option.

Select this option for example when your VLF application might be using `CALL_SERVER_FUNCTION` to run an RDMLX function or accessing an RDMLX file on the server.

If this setting is selected, the connection to the server uses the built in function `DEFINE_ANY_SERVER`. See the documentation for the restrictions for the built in function in the *Technical Reference Guide*.

This property is in the [Server Details](#) tab.

New Password

This password is used only for validating the user profile and password of users who must sign on to use the local database. It is not used for users who do not sign on, or for users who have the option of connecting to a server.

The administrator can use this field to assign default passwords for users that will be signing on to use the local database.

This property is in the [User Details](#) tab.

Physical File from which the Child Data Comes From

Specify here the name of the physical file that contains the data for the child object.

This is a [Program Coding Assistant](#) property.

Popup Panel Name

This is the name of your reusable part that will pop up when the user mouses over an instance list entry of this business object.

The reusable part must have as its ancestor VF_AC021.

In order for the pop up to be shown, the business object that displays the instance list must have the uListPopUp enabled, and the end-user must not have disabled instance list pop ups. The pop up panel can be any size you want. The panel is able to control its position via parameters on the uLoad method. See component DF_T3303 for an example.

This feature is only available for the primary instance list, for frameworks being run in Direct-X mode (Render mode M).

This property is in the Business Object [Instance List/Relations](#) tab.

Port Number

Specify the IP port number RAMP-TS sessions should use.

This property is in the [Server Details](#) tab.

Preferred web scheme/skin

The choice you make here affects the backgrounds used in any RAD-PADs you create while prototyping your application. Later you can choose to execute and deploy your Web browser application using one or more of these skins. Choose a preferred skin from:

Windows Classic Your browser application will be given an appearance that closely resembles the classic Microsoft Windows look of the platform it is executing on.

Web Page Your browser application will be given an appearance that distinctly identifies it as a web application.

Windows XP Your browser based application will be given an appearance that closely resembles the Microsoft Windows XP look, even when it is not executed on Windows XP platforms.

This skin is no longer available – please do not use.

This property is in the [Developer Preferences – Web Server](#) tab.

Private Definition/aXes Project Folder

Type in the name of the folder where to store the screens defined using the RAMP Tools. You must only type the folder name, not the path name. The folder will be created if it does not exist.

When not specified the private definition/aXes project folder defaults to /ts/screens and when specified it will become of subfolder of /ts/screens.

You should back up this folder on a daily basis.

You should only use private definition/aXes project folders to separate screen definitions on a completely separate and individual project basis.

You should never use private definition/aXes project folders to separate screen definitions on a developer, task or unit of work basis.

Screen definitions stored in separate private definition/aXes project folders cannot ever be merged with screen definitions stored in other private definition/aXes project folders.

Example

If your Private Definition/aXes Project folder is **/axes/ts/screens/MyProject/** this property would be completed as:

Private Definition/aXes Project Folder

This property is in the [Server Details](#) tab.

Private Working Folder

The private working folder is the folder where you will upload the files required to run the Visual LANSA Framework in a browser.

Before you upload the files, this folder must exist on your LANSA for the Web development system server and be accessible via HTTP.

We recommend you use the [Web Configuration Assistant](#) to create the private working folder.

If you want to create this folder manually, do this:

For a System i web server create a folder on the System i IFS. To make sure the folder is accessible via HTTP you can create it as a subfolder of the Lansa for the Web Images folder.

Type in the path to the private working folder.

For example, if your Lansa for the Web images folder is **LANSAIMG** and you created a subfolder of it called **VLF_Fred_Private**, the value you would enter into this field would be:

LANSAIMG/VLF_Fred_Private

Verify that the value entered is correct and that the folder is accessible by clicking the **Verify** button. The resulting Internet Explorer display should show the index of the folder.

If the verification process fails you should resolve the problem before proceeding. See LANSA for the Web Private Working Folder Problems for some ideas about problem resolution.

For a Windows web server create a folder on the Windows web server. You can create it as a subfolder of the Web Server's Images directory to make sure that it accessible via HTTP.

Type in the path to the private working folder.

For example, for an IIS web server, the Images folder has an alias name of Images. So suppose that the path for images was:

c:\Lansa\WebServer\Images

In the browser you can type `http://<nnn.nnn.nnn.nnn>/images/` to access the Images folder.

If you created a subfolder, for example,
c:\Lansa\WebServer\Images**VLF_Fred_Private**

the value you would enter into this field would be:

Images/VLF_Fred_Private

Verify that the value entered is correct and the folder is accessible, by clicking the **Verify** button. The resulting Internet Explorer display should show the index of the folder.

If the verification process fails you should resolve the problem before proceeding. See LANSAs for the Web Private Working Folder Problems for ideas about problem resolution.

Please note: each Framework developer needs to have a private working folder created for his or her exclusive use.

This property is in the [Developer Preferences – Web Server](#) tab. It is also a [Web Configuration Assistant](#) property.

Programmatic Identifiers for Building AKey and NKey Values

Specify the programmatic identification protocol you wish to use for this business object.

For example, for a "Products" business object you might decide to specify field PRODNO (Product Number) as the single programmatic identifier because the Product Number is enough to uniquely identify a "Product" instance to your programs.

Any field name can be specified in this list (even ones not currently defined in the LANSA data dictionary). The fields specified do not necessarily have to be columns in any physical file that you specify at the top of the form.

See [Visual Identifiers](#) and [Programmatic Identifiers](#) for more details of business object identification protocols.

This is a [Program Coding Assistant](#) property.

Property Type

Specifies the property type as Alphanumeric, Numeric or Boolean (TRUE/FALSE value).

This property is in the [Custom Properties](#) tab.

Prototype Only

You can execute the Framework in its original prototype mode by selecting this check box when you select the Framework file.

This option is only available when you execute the Framework as Designer.

In prototype mode all these Framework features default to using their original prototype state:

- The system IIP
- The bitmap and icon loader
- Filters
- Command handlers
- Snap-in instance list browsers

This property is in the Select Framework dialog.

·

RAD-PAD File Name

The name of the RAD-PAD file that is associated with the mock-up version of this filter.

This property is in the [Identification](#) tab.

RAD-PAD File Format

The recommended setting for this option is HTML format.

RTF format is a deprecated feature. You should not use it in new Frameworks.

This property is in the [Framework Details](#) tab.

RAMP Destinations

The RAMP destination screen name. For more information, refer to the RAMP documentation.

This property is in the [Commands Enabled](#) tab.

RAMP Javascript Charset

This property maps to the charset= attribute of a an externally included javascript file.

It specifies the character encoding to be used for the RAMP scripts file, for example vf_sy001_system_Nodes_RAMP_DEFAULT_SESSION.js.

This property is in the [Web/RAMP Details](#) tab.

RAMP Password

The password associated with the RAMP User.

Use of this option in end-user environments is unusual (refer to [RAMP User](#) for details).

When used, specify this option as a valid System i password, or use the special value *PROMPT to indicate that the user should be prompted for the correct password at the time that any RAMP connection needs to be established.

This property is in the [User Details](#) tab.

RAMP Tools Mode Load Path

Specifies the path that any RAMP-TS session started should use at design time when using the choreographer in the RAMP Tools. You should consult with your product vendor before changing this value.

This property is in the [Server Details](#) tab.

RAMP User

Applicable to RAMP applications, and only when the user profile that will be used by RAMP to connect to your System i server is different to the Framework user profile.

Use of this option in end-user environments is unusual.

Say you have defined user profile USERA to the Framework. Typically an end-user will start the Framework as USERA and by default any RAMP connection that needs to be established will also be started under user profile USERA. Only in the unusual situation where the end-user needs to start the Framework as USERA, but establish their RAMP session as, for example, USERB, would you use this option.

Specify this option as a valid System i user profile, or use the special value *PROMPT to indicate that the user should be prompted for the correct user profile at the time that any RAMP connection needs to be established.

This property is in the [User Details](#) tab.

Read Only

This is for tables which have no method for saving or updating data in their table data handler, or for those tables where the complete data set has been added and saved, and the designer does not wish to allow the administrator to modify the table data in future.

This property is in the Code Table [Definition](#) tab.

Referenced .NET Assemblies

Add here the .NET assemblies containing your snap-in components, one assembly in each line. You also need to add any dlls that are referenced in your assemblies such as LOpen.dll.

You can use either absolute or relative paths. If you use a relative path, the assembly has to be in the LANSAs partition execute folder.

This property is in the [Framework Details](#) tab.

Remember Key Values between Filter Executions

Check this option if the filter is to include code to remember the search criteria used between executions on the virtual clipboard.

For more information about the virtual clipboard see [The Virtual Clipboard](#).

This is a [Program Coding Assistant](#) property.

Relationship Handler

A relationship handler is an RDML function or reusable part that is called to dynamically expand the relationship between a parent and child object. By doing this you can improve filter performance by only adding root or parent objects to the instance list initially. For a sample of such handlers see the [Sample Relationship Handler Function](#) and the [Sample Relationship Reusable Part](#).

Relationship handlers are used by the shipped instance list only, they are not applicable when writing your own instance list.

This property is in the Business Object [Instance List/Relations](#) tab.

Relationship Type

Use this option to specify the type of relationship the selected business object has with the object being edited. The relationship can be:

- | | |
|----------------------|--|
| None | There is no relationship between the objects. |
| Loosely coupled peer | This business object can be displayed in the same instance list as the object being edited even though there is no hierarchical relationship between them. |
| Child or descendant | This object is a child object of the object being edited. |

This property is in the Business Object [Instance List/Relations](#) tab.

Report on Users - Imbedded Interface Point (Id)

This is the name of a reusable part that is executed when the Administrator presses the User Authorities Report File button in the User Details tab (accessed using the Users option of the Administration menu).

The specified reusable part requests the reporting options from the administrator, and then produces a report.

The report could be in the form of a .csv file, or it could be written out to a database. It is entirely up to the programmer to choose:

- The options requested from the administrator when running the report
- The report structure and content
- The way in which the report is output.

If the property is left blank, the standard report program is used, which outputs a .csv file (which can be read by excel.)

This feature is intended for developers who do not want to use the standard report program, because they want different report content/structure/summaries, or have exceedingly large volumes of user/authority data, or want to output the report data in a different form (e.g. to a local database).

See the source of the shipped example component UF_REPUS for a working example of a component like this.

Note the ancestor of the component must be VF_AC022.

It is suggested that developers start by comparing this example program with its output, and then make a copy and modify their program to get the results they want.

This property is in the Framework [User Administration Settings](#) tab.

Restricted Access

This property overrides the normal behaviour of Framework objects which is to inherit their use authority from their parent.

Normally, for example, if a user has authority to use an application, they will automatically have authority to use a business object in that application. But if the business object has Restricted Access checked, then it will not inherit USE authority from its parent application. The administrator will have to specifically authorize the user to that business object to allow the user to use it.

In most cases Restricted Access should be left unchecked, so that authorities specified at a higher level will cascade to all child objects. This makes authority specification easier.

There is one case however, where you may find it useful to check Restricted Access. Suppose you have already set up a Framework, all your users, and their authorizations. Now you wish to add a new object (application or business object or command ref) to the Framework. For this new object you do not want users who are authorized to its parent to automatically get authority to the new object. In this case you would check the new object's Restricted Access property.

This property is in the [Commands Enabled](#) tab and the [Identification](#) tab.

Routine to listen for signals to update the instance list

Select this option if you want to create filters that listen for changes from RAMP command handlers.

For more information, refer to the RAMP Guide.

This is a [Program Coding Assistant](#) property.

Save and Restore Instance Lists

Select this option if you want the instance list saved when the Framework is closed and restored the next time the Framework is executed.

In this way the user does not have to use the filters every time they start up the Framework.

This option is only applicable to **Windows** applications.

This property is in the Business Object [Instance List/Relations](#) tab.

Save as Deployment Server

Select this option if you want these server details be used when deploying the application. The characteristics of this server will then be used as default values in the deployed application.

For example, if you start a deployed VLF-WEB application that uses RAMP-TS it needs to know the IP address of your RAMP-TS server to be able to initiate the RAMP 5250 session.

The IP address it uses can come from one of two places:

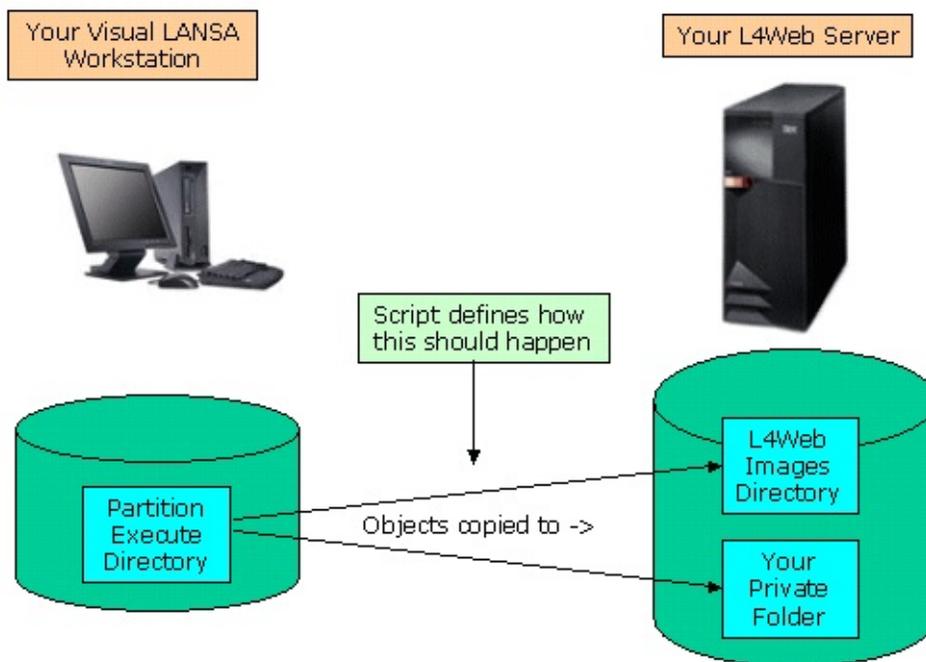
1. An IP address you specify on the start-up URL
2. The default IP address - which was saved from the server definition with "Save as deployment server" ticked when the framework definition was last saved.

A specific IP address on the start up URL will always take precedence.

This property is in the [Server Details](#) tab.

Script for Uploading to your LANSA for the Web Folders

The Framework needs to copy data from your Visual LANSA workstation to the LANSA for the Web Images folder and to your LANSA for the Web Private Working Folder.



It can perform these copies using FTP or simple DOS COPY commands (providing that you have a mapped drive to the your IFS folders).

It does this by expanding and then executing a small skeleton script that you provide.

You need to create (and sometimes modify) the script that will be used.

If you have a mapped drive and have input the Optional Mapped Drives to your LANSA for the Web Folders details in step 10:

- Change the Script Type to "DOS File COPY Commands"
- Click the Generate Example Button.
- A script example should appear in the script window.
- The script window is input capable and you can change the script if you need to.
- Review the script and check that the directory names are correct. Change them as required so that information can be copied from your partition execute

directory to your LANSA for the Web images directory and your private folder.

If you do NOT have a mapped drive, or prefer to use FTP file transfers:

- Change the Script Type to "FTP File Transfer Commands"
- Click the Generate Example Button.
- A script example should appear in the script window.
- The script window is input capable.
- Replace the "===> Replace this line with your FTP user profile <=== " string in line 2 of the script with the user profile you use for FTP file transfers. Typically this is your System i user profile (eg: QPGMR).
- Replace the "===> Replace this line with your FTP password <=== " string in line 3 of the script with the user profile you use for FTP file transfers. Typically this is your System i user profile password (eg: PGMR). Note that your user profile and password details are not stored in the Framework XML file and are only ever kept in your Visual LANSA system.
- Review the script and check that the directory names are correct. Change them as required so that information can be copied from your partition execute directory to your LANSA for the Web images directory and your private folder.

For example, after pressing the "Generate Example" button you might see this:

```
OPEN 608.89.58.81
===> Replace this line with your FTP user profile <===
===> Replace this line with your FTP password <===
LCD "C:\PROGRAM FILES\LANSA\X_WIN95\X_LANSA\x_dem\execute"
PROMPT
..... etc .....
```

If your FTP user profile was QPGMR (say) and your password was MOUSE (say) then you should change the generated example to be like this:

```
OPEN 608.89.58.81

QPGMR

MOUSE
```

```
LCD "C:\PROGRAM FILES\LANSA\X_WIN95\X_LANSA\x_dem\execute"  
  
PROMPT  
  
..... etc .....
```

For a Windows web server and for an Apache web server running on System i, you may need to change directory names.

For example, if the virtual name for your Images folder was Images after pressing the "Generate Example" button you might see this:

```
CD "//Images"  
  
MPUT <<PUT_SHARED_FILES>>  
  
CD "//Images/My_Private_Folder"  
  
MPUT <<PUT_PRIVATE_FILES>>  
  
..... etc .....
```

If the real name for your Images folder was LANSAIMG you should change the generated example to be like this:

```
CD "\lansaimg"  
  
MPUT <<PUT_SHARED_FILES>>  
  
CD "\lansaimg\My_Private_Folder"  
  
MPUT <<PUT_PRIVATE_FILES>>  
  
..... etc .....
```

Note that the upload script is static.

If you change any of your folder or mapped drive details you will probably need to regenerate the upload script again.

This property is in the [Developer Preferences – Web Server](#) tab.

Search Field Width

Applies to **Windows** Framework and **.NET** applications.

This option defines the width of the search field on the tool bar. A value of zero indicates the search field should not appear and the search feature should not be enabled.

The search field only appears in the main Framework window or in full Framework window(s) opened from within it. The search list is a statically positioned list. If you resize the VLF main window while it is displayed, it will stay in the same position. When the user moves the focus away from the search field, the search list will immediately disappear.

The most recently used business object details are stored in the system virtual clipboard, so they are lost when the virtual clipboard is cleared.

The first time that a user performs a search a list of text from all authorized business objects is built. This process checks the user's authority to use the business objects, so if you are using an expensive business object authority checking IIP you should consider this when enabling this feature.

The words that appear in the search field and the results list are all exposed as normal customer modifiable multi-lingual variables. They may be customized in the usual manner.

Also see [Quick Find Box on the tool bar](#) and [Allow Search/Recently Used Limit](#)

.

This property is in the [Framework Details](#) tab.

Select the Keys of the Selected View to be used for Search Operations

Select the key(s) of the selected view that are to support searching.

Imagine that you had two views of an employee's table available:

- View A was keyed by DEPARTMENT (Department), SECTION (Section) and EMPNO (Employee Number).
- View B was keyed by SURNAME (Employee Last Name) and GIVENAME (Employee Given Name).

Now imagine that you wanted your end-users to be able to:

- Search for all the employees that work in a specific department or section.
- Search for employees by full or partial name.

To do this you would probably create 2 filters:

The first (1) would use view A with these options

Keys to be used in search: DEPARTMENT and SECTION only.

User must specify all chosen keys: No. Just a DEPARTMENT value alone may be specified to create a list of all the employees in a department.

Allow generic searching: No. Searching for departments or sections with generic names has no business value and may be confusing.

The second (2) would use view B with these options

Keys to be used in search: SURNAME only.

User must specify all chosen keys: Yes. At least one letter of a surname must be specified before the search button can be clicked to prevent extremely large searches being done by accident.

Allow generic searching: Yes.

Note: Key selections need to be contiguous from the first key. For example, if you select key number 3 you must also select key numbers 1 and 2.

This is a [Program Coding Assistant](#) property.

Select the View to Be Used For Filtering and Searching Operations

Select the view of the primary physical file that is to be searched by this filter.

This is a [Program Coding Assistant](#) property.

Selection Block Size

This is the block size parameter used by the `CONNECT_FILE` Built-In function when connecting to the server. This property defines the number of records that are transferred, in one hit, from the server by `SELECT` commands executed on the client system. Performance is improved by using large block sizes, so the default when you define a new server is 500. Conversely, using small block sizes degrades the performance of your application.

It is recommended that you do not set this to a value less than 50.

You can override this value in any program by use the `CONNECT_FILE` built-in function.

You can do this for all files, generic groups of files or individual files.

Refer to the `CONNECT_FILE` built-in function in the LANSAP Technical Guide for more information.

Note that if your code performs `UPDATE` or `DELETE` operations without `WITH_KEY` or `WITH_RRN` parameters, the operations are performed on the last record in the block, which may not be the the last record read into your program.

Therefore, if you must use `UPDATE` or `DELETE` operations against the last record read, it is recommended that you supply a `WITH_KEY` or `WITH_RRN` value instead, or temporarily use the `CONNECT_FILE` built-in function to push the block size down to 1, then restore it back to a high value.

This property is in the [Server Details](#) tab.

Selection Limit

This is the Selection limit parameter used by the CONNECT_FILE Built-In function when connecting to the server. It sets the maximum number of records that can be read from the Server.

For more information refer to the LANSAP Technical Reference guide and review the help for the CONNECT_FILE Built-In function.

This parameter can be overridden using your own logic. See [Imbedded Interface Points \(IIPs\)](#).

This property is in the [Commands Enabled](#) tab and the [Server Details](#) tab.

Sequence

Use this property to assign a sequence number to this object. This number controls the position of this object relative to other objects of the same type. For example, an application with a sequence number of 1 is displayed before an application with a sequence number 2 in the application bar.

If you do not specify a sequence number, the objects are displayed in alphabetical order of their captions.

For custom properties this is the sequence in which the property is to be displayed to the administrator (within the Framework, application or business object to which it belongs).

This property is in the [Custom Properties](#) tab and the [Identification](#) tab.

Sequence Using

Use this option to select which of the table fields is to be the one used to determine the sequence of the entries in either the combo box or the list of radio buttons.

If you choose the option "All key fields" this means that the table entries will be displayed in the order they arrive from the table data handler function.

If the table data handler function is the generic table data handler then the arrival sequence is sorted by the key fields.

This property is in the Code Table [Visualization](#) tab.

Server Client Translation Table

For System i servers only specify the server to client translation table to be used.

Note that these values are automatically defaulted from the client partition definition.

The default translation table used for Framework server definitions using RDMLX partitions/connections is *JOB.

Ensure that your client and server partition definitions match.

This property is in the [Server Details](#) tab.

Server IIP function to validate sign on

This is the name of a lansa IIP function that runs on the server during sign on. It is run after the user has connected to the server, but before VLF authority is evaluated. It receives the user profile that the user signed on as, and can return a user profile, as well as other values.

This function must exist on the server.

See the source for function UF_SYSBR/UFU0005 for more details

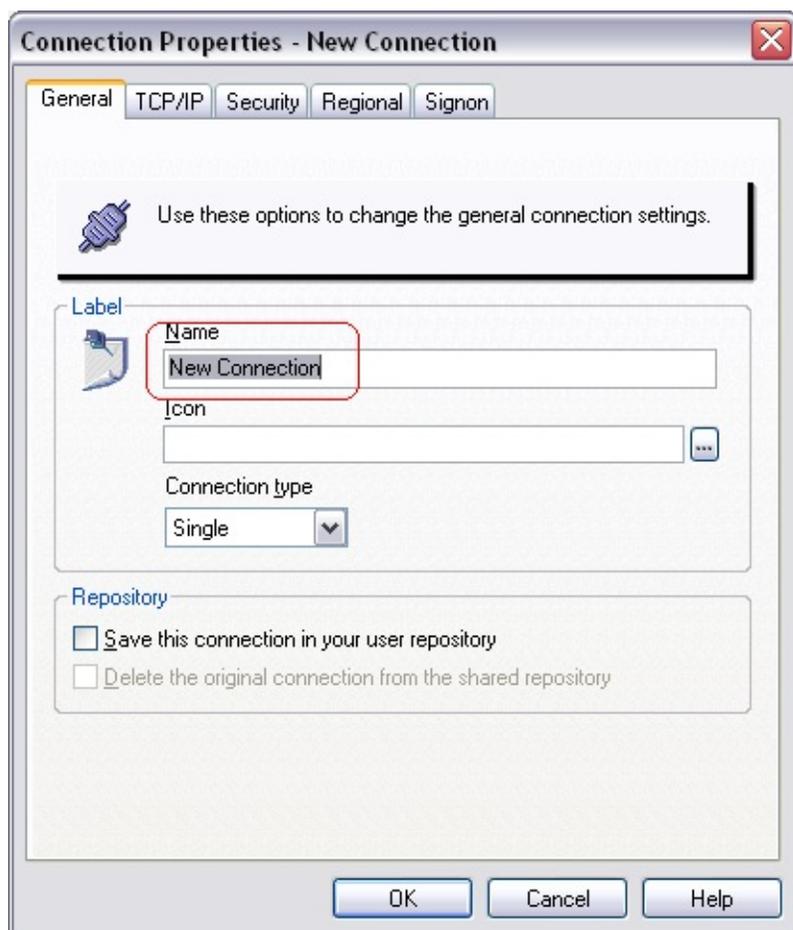
This property is in the [Server Details](#) tab.

Server Name

Enter here the identifying name of the server.

For other types of server except newlook, this name should be the same as the Partner LU Name you used when you defined the server to the LANSAC Communications Administrator.

For newlook servers, enter the name of the newlook connection as defined in the newlook Connection Properties panel.



You can leave this field blank in which case the IP address and Port will be used.

If you leave all the fields blank, the newlook connection panel will be displayed when the Framework is trying to establish a connection.

This property is in the [Server Details](#) tab.

Server Overrides

The X_RUN exceptional arguments may be used to override the parameters used on the X_RUN command started on the server system:

By default, the following client X_RUN parameter values are passed to (and inherited by) the X_RUN command started on the server system: LANG=, PART=, DEVE=, DBID=, DBII=, DBUT=, DBIT=, DBUS=, PSWD=, USER=, PSPW=, CMTH=, CDLL=, TPTH=, RPTH=, DATF=, XAFP=, PRTR=, HSKC=, ODBI=, TASK=, PPTH=, INIT=, TERM=, ITRO=, ITRL=, ITRM= and ITRC=.

All other X_RUN parameter values on the server system are defaulted (on the server system) in the usual manner (i.e.: from a profile file, from system environment settings, etc). Refer to the definition of the X_RUN command for details of all parameter values and the methods by which they can be specified and defaulted.

You may override any server X_RUN parameter (via the X_RUN exceptional argument value) except for CMTH=, CDLL=, PROC=, MODE=, PART=, LANG=, DEBUG=, DEVE=, DATF=, XAFP=, USER= and PSPW=. These X_RUN arguments are unconditionally inherited from the client system.

Override parameters may be given a specific value, or the special value *SERVER, which indicates that the server default should be used. As an example, a Windows client using DBII=*NONE might connect to a Windows Server running SQL Server. If, by default, Windows uses the database type SQLANYWHERE, DBUT needs to be overridden. The X_RUN exceptional argument value could be set to either DBUT=MSSQLS or DBUT=*SERVER.

This property is in the [Server Details](#) tab.

Server Settings XML File

Use the Server Settings XML File property to specify the name of the file where server settings for the Framework are stored. This file is located in the Execute directory of the current partition.

The default value is Vf_Sy001_Servers.xml.

Please refer to [Development Architecture](#) before changing this value.

This property is in the [Framework Details](#) tab.

Server Type

Select the type of the server to be defined.

When defining a RAMP-TS server, typically you would select the option LANSA for System i + RAMP-TS. Similarly, when defining a newlook server, typically you would select the option LANSA for System i + RAMP-NL.

If your System i and RAMP-TS or newlook servers have different IP addresses (even though they might be the same physical server), choose the RAMP-TS Only or RAMP-NL Only option.

This property is in the [Server Details](#) tab.

Shortcut

Commands that appear in menus may have an associated short cut key. Using the short cut key is equivalent to clicking on the command in the menu. The list of available short cut key combinations is shown in the drop-down.

Note: The preferred shortcut keys are function keys and Ctrl+letter combinations. It is possible to set the shortcut key to Alt+letter, but this is not recommended because as a rule Alt+letter combinations are used as the access key based on the underlined letter of the caption.

F1, Ctrl+F1 and Ctrl+Shift+F1 will always bring up the help text of your application.

This property is in the [Identification](#) tab.

Show Additional Columns

Use this option to indicate whether any additional columns for this business object should be displayed in the instance lists selection tree.

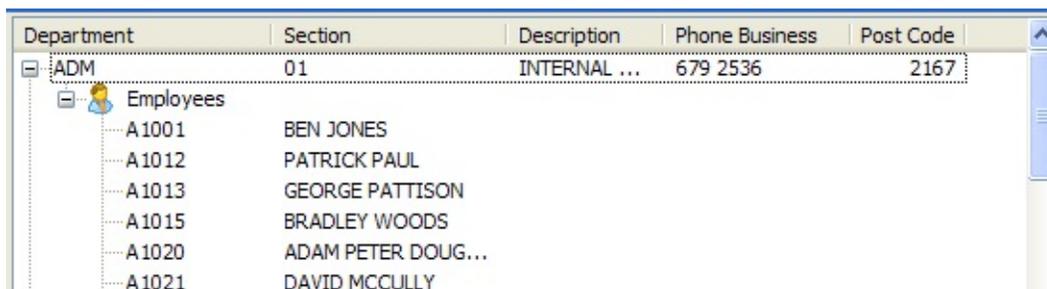
In this example the option has been selected so that additional columns from the SECTIONS and EMPLOYEES business objects are to appear in the main instance list tree together:



Department	Section	Description	Phone Business	Post Code
ADM	01	INTERNAL ...	679 2536	2167
Employees				
A1001	BEN JONES	798 0543	144 Frog	2002
A1012	PATRICK PAUL	222 2222	6 Camillo Avenue	2147
A1013	GEORGE PATTISON	212 3569	12 Augusta Av...	2016
A1015	BRADLEY WOODS	789 4562	59 Darley Road,	2030
A1020	ADAM PETER DOUGLAS	639 5188	6 Reading Ave...	2147
A1021	DAVID MCCULLY	159 6845	15 Baker Place,	2153
A1025	MARY ROBINSONNNN	456 1852	14 Whitby Road,	2005

Notice how the additional columns from the SECTIONS business object and the EMPLOYEES business object both appear in the tree, even though they contain quite different content? When you use this option there should be some degree of commonality in the way that the different business objects use the additional columns in the tree.

If this option was unchecked, then the display would change to this:



Department	Section	Description	Phone Business	Post Code
ADM	01	INTERNAL ...	679 2536	2167
Employees				
A1001	BEN JONES			
A1012	PATRICK PAUL			
A1013	GEORGE PATTISON			
A1015	BRADLEY WOODS			
A1020	ADAM PETER DOUG...			
A1021	DAVID MCCULLY			

The SECTIONS additional columns appear in the tree, but now only the most basic EMPLOYEES details now appear. Note that in this type of display the column headings and layout details come from the parent business object, in this case SECTIONS.

This property is in the Business Object [Instance List/Relations](#) tab.

Show Current Business Object in Window Title

Check this option to force the Framework to always show the current business in window titles.

If this option is unchecked, the Framework caption will be used for all window titles unless the Show the 'Windows' Menu in this Framework option is checked in which case all windows will show the current business object as the window title regardless of this option.

This property is in the [Identification](#) tab.

Show in Help About Text

If this option is checked, the Framework version number will be automatically included at the start of the text displayed when end-users the Help About options. This allows you to identify which version of your Framework an end-user is actually executing.

This property is in the framework's [Identification](#) tab.

Show in Menu

Choose the menu in which the command is displayed.

If you do not include the command in any menu, it will be displayed in the pop-up menu associated with the object for which it is enabled. However, it is good practice to include commands in the menu bar because that is where many users first look for a command.

To create a new menu, use the Menus... option of the Framework menu.

Note that the Framework and Administrator menus are built in the Framework and cannot be changed. See [Menu Options in Brackets](#) for information of how to run the Framework with these menus hidden.

This property is in the [Toolbar and Menus](#) tab.

Show When Disabled

This option only applies when the command is set to appear on a menu in the main menu bar. It does not apply to command items appearing on pop-up menus.

Use this option to specify whether the menu option for this command is shown when the command is not enabled for the selected object.

This option applies to **Windows** and **Web** Framework applications.

This property is in the [Toolbar and Menus](#) tab.

Show on Instance List Tool Bar

Select or unselect this option to control whether a button for a command reference should appear on the instance list tool bar associated with the instance list.

Note: If child or parent business objects have a reference to the same command definition, a button for these references may still appear.

This property is in the [Commands Enabled](#) tab.

Show On Popup Menus

Select or unselect this option to control whether this command reference should appear in right mouse button popup menus within the Framework.

Typically you would not select this option when:

- The object with which the command is associated has a single default command.
- You want the command to only be invoked from the menu bar or toolbar.

This property is in the [Commands Enabled](#) tab.

Show on Toolbar

Select this option to show this command in a toolbar button. The image on the button is the bitmap associated with the command.

Note that the image shown on the command handler tab for this command is the icon associated with the command, so you should try to use the same or similar images for both the bitmap and the icon.

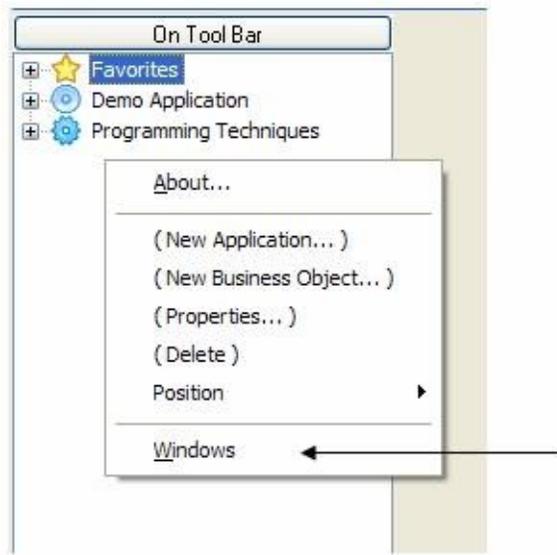
This property is in the [Toolbar and Menus](#) tab.

Show the 'Windows' Menu in this Framework

Use this option to show or hide the Windows menu option on the Framework menu bar:



And on right mouse pop-up menus:



The Windows menu option allows secondary windows containing instances of the entire Framework, a specific application, a specific application view or a specific business object to be opened for concurrent use.

Being able to have multiple windows open at the same time allows end-users to more easily perform and seamlessly interrupt concurrent tasks within your application.

The number of concurrent windows is set using the [Number of Additional Windows a User can have Open Concurrently](#).

You may have designed dependencies into your filters or commands handlers that preclude them from functioning correctly when there are multiple instances open in multiple windows. In this case use the [Allow this Object to be Opened in a New Window](#) option to prevent them from being used in multiple open windows.

Be aware that the Overall Theme options are also on this menu and if users are

allowed to change the Overall theme of the Framework this menu should be shown.

The Windows menu option is not applicable to Web Browser applications.

This property is in the [Identification](#) tab.

Show on Connect Dialog

VLF.WIN only.

This property controls whether a server appears in the list of servers on the end-user's Connect dialog.

This property is in the [Server Details](#) tab.

Show When Disabled

Use this option to specify whether the toolbar button for this command is shown when the command is not enabled for the selected object.

This option only applies to **Windows** Framework applications.

This property is in the [Toolbar and Menus](#) tab.

Snap in Instance List Browser ID

Specify here the identifier of your own instance list for your Windows Framework application which will replace the standard shipped instance list.

Your instance list must be a compiled reusable part.

The instance list typically appears in the upper right of the Framework and it displays the list of business object instances currently selected.

See [Optionally Create Your Own Instance List](#) for more details.

This property is in the Business Object [Instance List/Relations](#) tab.

Specify the Underlying Physical File that Will Be Searched by this Filter

Specify the name of the primary physical file (i.e.: table name) that will be searched by this filter.

This is a [Program Coding Assistant](#) property.

Options

This option specifies what is displayed in the large right-hand frame when the Framework or application starts up. The options are:

- | | |
|--------------|--|
| None | No action, the introduction panel simply hides itself from view. |
| Full Image | The image specified as Bitmap is displayed so as to fill the entire viewing area of the introduction panel. The URL property has no meaning when this option is selected. |
| Real Image | The image is displayed in its real size, centered in the viewing area on the introduction panel. If a URL value exists then its text is centered below the image so that it can be clicked on. |
| No Image | No image is to be displayed. If a URL value exists its text is centered in the viewing area. If no Introduction URL exists a completely blank form is displayed. |
| Embedded URL | The URL specified is immediately displayed as a web page imbedded in the introduction panel. |

This property is in the Framework [Startup](#) tab.

Stay Active

Set this option to YES to indicate that the filter or command handler should stay active even when it is not visible.

This option is recommended for heavily used filters and command handlers. Using it usually sacrifices memory usage for a performance increase.

Alternatively, setting this option to NEVER can minimize memory usage. The NEVER Stay Active option indicates that **when you move between business objects**, any inactive filters or command handlers (i.e. ones that are on tabs that are not visible to the user) should be terminated. This means that their `uTerminate` methods should be executed. (Note: Visible/active filters or command handlers cannot be terminated because the user may redisplay them at any time).

DEFAULT sets the Stay Active option for filters and command handlers to the value set on the new Framework level option, Stay Active Default for Command Handlers and Filters.

Programming notes:

Normally a filter or command handler may be destroyed when the tab is no longer required and recreated when it is displayed (the random creation/destruction process is controlled by the system, not the Framework). If a filter or a command handler is heavily used, this process can be costly and can be avoided by setting Stay Active to YES.

If you have filters or command handlers that are specified as Stay Active, you can find out when the filter is being created for the first time (`uInitialize`) and when it is being destroyed (`uTerminate`) so that you can in turn yourself create and destroy things that your filter and command handler uses.

If you use `uInitialize` and `uTerminate` with a non-stay active filter or command handler they still work, but they may be invoked frequently as the filter or command handler is being created and destroyed, and they do not run every time the object is hidden and displayed.

This option only applies to **Windows** Framework applications.

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

Stay Active Default for Command Handlers and Filters

This option defines the default value of the Stay Active option for all command handlers and filters that have their Stay Active option set to DEFAULT.

Set this option to YES to indicate that filters or command handlers should stay active even when it is not visible.

This option is recommended for frameworks with many heavily used filters and command handlers. Using it usually sacrifices memory usage for a performance increase.

The NEVER Stay Active option indicates that when you move between business objects, any inactive filters or command handlers (i.e. ones that are on tabs that are not visible to the user) should be terminated. This means that their uTerminate methods should be executed. (Note: Visible/active filters or command handlers cannot be terminated because the user may redisplay them at any time).

The NO setting means that the framework does not attempt to control the termination of filters and command handlers.

This property is in the [Framework Details](#) tab.

Store Users in XML File and Store users in DBMS Tables VFPPF06/07

The Visual LANSA Framework is shipped with a user profile and authority management facilities built in.

These facilities can be enabled or disabled (see [Use Framework Users and Authority](#)).

If they are enabled the user and authority details need to be stored:

- In an XML formatted file, or
- In database tables VFPPF06 and VFPPF07.

The option to use XML formatted files should only be used in standalone Windows systems.

In all other situations storage in database tables is the recommended option.

Web browser applications must use the database tables option.

When using an XML formatted file:

- The default name is Vf_Sy001_Users.xml.
- The default location is the Execute directory of the current partition.
- If the XML file needs to be in a different directory specify the full path as well as the file name.
- Please refer to [Associated XML Definition Files](#) before changing this value.

When using DBMS tables:

The DBMS tables should be located on your server system.

When updating user profile details you need to be connected to the server.

This property is in the Framework [User Administration Settings](#) tab.

SubTypes

Business objects may optionally have a SubType associated with them.

For example, a business object named BankAccount might be sub-typed as being a Savings, Check or Investment Account.

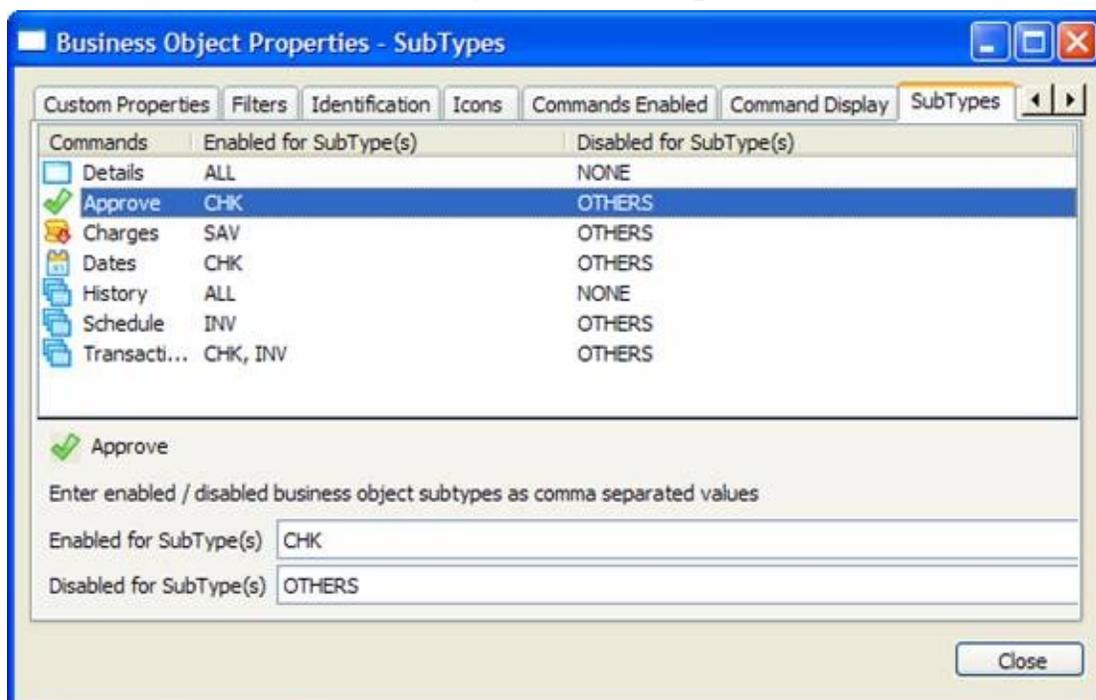
The purpose of subtypes is to allow the display of commands (and their associated tabs) for the business object to be conditioned so that they are only visible and useable for certain subtypes.

For example, the command/tab named Transactions might only be validly displayed for Check and Investment accounts.

Likewise, the command/tab named Charges might only make sense when used with a Savings account.

Subtypes are represented by a code that you can associate with a business object instance. For example you might choose the codes SAV, CHK and INV for the 3 BankAccount subtypes.

You specify how commands and subtypes are related by using the SubTypes tab associated with the business object. For example:



SubTypes should be no more than 5 characters long, and contain uppercase

letters of the English alphabet (A ->Z) or numbers (0 -> 9) only.

The values ALL, NONE, ALLOTHERS and OTHERS should not be used for SubTypes.

SubTypes are only applicable to instance level commands/tabs.

Any Command that is not enabled for all subtypes needs to have its Default Command option set to 'Never'

Once you start using subtypes for a business object instance list you should use them for every instance list entry.

When you insert or update an entry into an instance list you may optionally specify a subtype to be associated with the entry.

Examples In VL Components and WAMs

```
Case #T_TYP
```

```
when (= SAVINGS)
```

```
  Invoke Method(#avListManager.AddtoList) AKey1(#T_Acc)  
  VisualID1(#T_Acc) Visualid2(#T_Nam) AColumn1(#T_Typ)  
  NColumn1(#T_BAL) SubType(SAV)
```

```
when (= CHECK)
```

```
  Invoke Method(#avListManager.AddtoList) AKey1(#T_Acc)  
  VisualID1(#T_Acc) Visualid2(#T_Nam) AColumn1(#T_Typ)  
  NColumn1(#T_BAL) SubType(CHK)
```

```
when (= INVESTMENT)
```

```
  Invoke Method(#avListManager.AddtoList) AKey1(#T_Acc)  
  VisualID1(#T_Acc) Visualid2(#T_Nam) AColumn1(#T_Typ)  
  NColumn1(#T_BAL) SubType(INV)
```

```
Endcase
```

Other Examples

Working examples of SubTypes are shipped in the Programming techniques Application (Advanced examples). Refer to the shipped example VL components DF_T2801/02 and WAMs DM_T2801/02.

These properties are set in the Business Object [Subtypes](#) tab.

System Info

This option determines where the Microsoft program that provides information about the currently running Windows system is located.

You can specify these options for this value:

Blank If this property is blank the system information button will be invisible.

***Default** If you specify *DEFAULT - the Framework will use:

```
Use Builtin(GET_REGISTRY_VALUE) With_Args(HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\SystemInfo\Path)
```

to get the path to the windows program that displays system info.

Any other value Any other value will be the path used to find the windows program that is appropriate for the PC).

This property is in the [Help About](#) tab.

Table Definitions XML File

Use the Table Definitions XML File property to specify the name of the file where code table definitions for the Framework are stored. This file is located in the Execute directory of the current partition.

The default value is Vf_Sy001_Tables.xml.

This property is in the [Framework Details](#) tab.

Technical Support

Enter here the address of the web page where the user can get technical support.

When specified, URLs should always be prefixed by http://, https:// or special value *auto*.

At execution time in VLF.WIN applications *auto* is replaced by the current LANSAPARTITION's execute folder name, and in VLF.WEB applications *auto* is removed from the URL.

For example, *auto*MYFILE.HTM would become C:\Program Files\LANSA\X_WIN95\X_LANSA\x_dem\execute\MYFILE.HTM in VLF.WIN applications and MYFILE.HTM in VLF.WEB applications.

This property is in the [Help About](#) tab.

Temporary Directory on PC

The temporary directory is used internally to store temporary error reports for emailing and for storing instance lists between Framework executions.

If left blank, the temporary directory defaults to the normal system defined temporary directory. Usually there is no need to change this default.

On CITRIX metaframe systems the system defined temporary directory concept can cause problems, and therefore it may be necessary to define different temporary or work directories for different users.

This property is in the [User Details](#) tab.

Temporary Folder Name

The Web Configuration Assistant creates a temporary folder named VLF_temporary_files on the web server.

The temporary folder is used to store temporary state information for active Web browser Visual LANSA Framework applications.

See [What is a temporary directory and what is it used for?](#)

This is a [Web Configuration Assistant](#) property.

The location of the images directory on the server

This is the full path of the images directory on the webserver. In other words this is the location of the images directory as you would find it using Windows Explorer if you were signed on to the webserver.

Normally this property is automatically detected. If you are required to specify this property, you will need to be able to view the webserver to determine where the images directory is.

A typical example of a location on a Windows web server would be:

C:\PROGRAM FILES\LANSA\WEBSERVER\IMAGES\

A typical example of a location on an iSeries web server would be:

/LANSA_DC@PGMLIB/webserver/images/

This is a [Web Configuration Assistant](#) property.

The physical file that most closely resembles this business object is

Many business objects have a database file / table that closely resembles them. For example the CUSTMST table may closely represent the business object "Customers" and the table PRODMST may closely represent the business object "Products".

If you specify a file name the Program Coding Assistant will attempt to automatically derive an identification protocol for you. Refer to [Visual Identifiers](#) and [Programmatic Identifiers](#) for more details about business object identification protocols.

The method used to derive an identification protocol works like this:

- The programmatic identifiers are set to be the same as the primary key of the table.
- The visual identifiers are set to be the same as the primary key of the table.
- While the visual identifier 2's aggregate length is < 80 the first 2 real or virtual fields in the table definition with a length >= 15 are added to the visual identifier 2 definition.

The derived identification protocol is of course only a "guess" on the part of the Program Coding Assistant and you should always review and alter it to be exactly what you want.

This is a [Program Coding Assistant](#) property.

The type of server

The Web Configuration Assistant detects the type of server you are configuring.

This is a [Web Configuration Assistant](#) property.

The webserver's images path

This is the images path that a browser looks for on the webserver:

<http://ip:port/Images>.

This is a [Web Configuration Assistant](#) property.

Timeout to use for developers

If you are using a web browser Framework application as a developer (developer=Y in the URL) then a check for failing filters or command handlers will be activated. If a filter or command handler fails to respond in a specified time a message like this will appear:



You can then wait for the filter or command handler to complete, check for error information on the web server or shut down the Framework.

Use this option to set the timeout to be used in error resolution for developers. It can be in the range of 1 to 3600 seconds. The default is 20 seconds.

This property is in the [Web/RAMP Details](#) tab.

Toolbar Button Caption

You can optionally create a caption for the toolbar button. You will probably need to increase the [Toolbar Button Width](#) to make the caption fit properly.

This property is in the [Toolbar and Menus](#) tab.

Toolbar Button Width

The width of the button in pixels. You may need to adjust this value to make the image or the caption on the button fit properly.

This option only applies to **Windows** Framework applications.

This property is in the [Toolbar and Menus](#) tab.

Toolbar Group

You can optionally group the buttons in the toolbar. The groups are distinguished by a separator.

To create a group assign it a number in this field.

This property is in the [Toolbar and Menus](#) tab.

Tool Bar Height

Use this option to set the height of the tool bar. In this way it is possible to create a tool bar that has more than one row of buttons.

This option only applies to **Windows** Framework applications.

This property is in the [Framework Details](#) tab.

Tool Bar Style

Tool Bar Style can be Advanced or Simple.

Advanced toolbar generally look like this example, using the command bitmaps and a small size.

Use the advanced style when a large number of tool bar options are required and they need to wrap onto secondary lines:



Simple toolbars present a simple and easier interface for new users. They are built from the command icons and generally are much larger, like this example.

They do not wrap on secondary lines so the number you can display is limited by the width of the device.



This property is in the [Framework Details](#) tab.

Trim Working Set

Set this property to reduce the amount of memory your Framework uses.

If this option is enabled in a VLF.WIN application the working set of the windows process executing the Framework is trimmed:

- When the start up of the Framework is complete
- When the main Framework window has been minimized for approximately 20 seconds.

This property is in the [Framework Details](#) tab.

Type of Layout Style to be Used

Note: WEBEVENT command handlers and filters are a deprecated feature. Do not use them for new work.

Specify the type of layout to be used for this command.

This option only applies to WEBEVENT command handlers and filters.

This property is in the [Commands Enabled](#) tab the [Filter Snap-in Settings](#) tab.

Unique Identifier

This is a unique internal identifier automatically generated by LANSAs to uniquely identify this object. The identifier is numeric and may change between Framework executions.

This property is in the [Identification](#) tab.

Update File

Use this property to specify the newlook update program file to be used when the application is deployed.

References to the nlupdate.txt file must specify the subdirectory in which it resides on your web server (which is below the directory containing the VF_SY120.js file).

Deployment options	
Update File	ramp_newlook/nlupdate.txt
CODEBASE	RAMP_NewLook/newlook.cab#version

Alternatively, you can specify this value as a URL parameter when starting your application:

```
+NLUPDATEFILE=
```

This property is in the [Server Details](#) tab.

Uppercase

For alphanumeric custom properties check this box to ensure that the property values are always presented and input in uppercase.

This property is in the [Custom Properties](#) tab.

Upper and Lower Case Password

This setting is applicable to iSeries servers only. If checked, the password entered by the user is validated exactly as typed. If unchecked (the default value), the password is always uppercased before being compared with the stored value.

For servers other than iSeries, the password entered by the user is always validated exactly as typed.

This property is in the [Server Details](#) tab.

URL

Specifies the URL of a web page that is to be shown at startup. (e.g. <http://www.lansa.com>).

When specified, URLs should always be prefixed by `http://`, `https://` or special value `*auto*`.

At execution time in VLF.WIN applications `*auto*` is replaced by the current LANSA partition's execute folder name, and in VLF.WEB applications `*auto` is removed from the URL.

For example `*auto*MYFILE.HTM` would become in Framework Windows applications:

```
C:\Program  
Files\LANSA\X_WIN95\X_LANSA\x_dem\execute\MYFILE.HTM
```

And in Framework web applications:

```
http://<your host>/images/VLF_Private_folder/MYFILE.htm
```

This property is in the Framework [Startup](#) tab.

Use a Grid for Displaying Any List

If you have specified fields that you want to appear in a list at the bottom of your command handler then check this box if the list type to be used is a grid.

If you uncheck this box then a list view will be used.

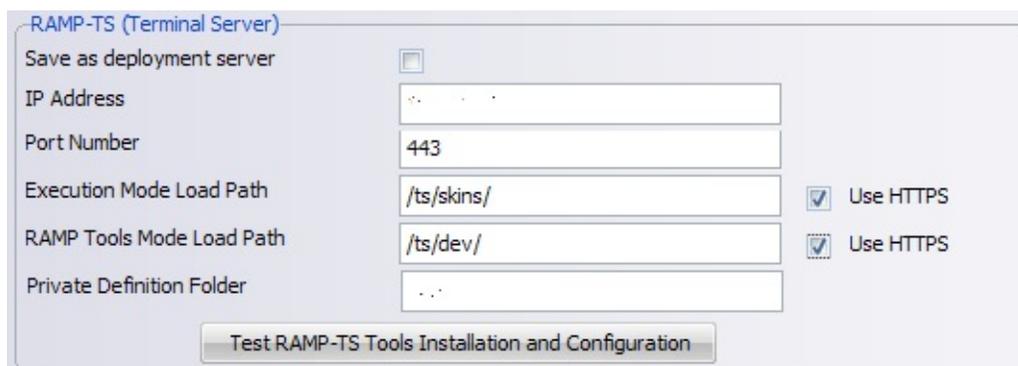
This is a [Program Coding Assistant](#) property.

Use HTTPS

Select this option to launch RAMP-TS using a URL with the https:// prefix instead of the usual http:// in order to connect to an aXes application server with Secure Sockets Layer (SSL) activated.

If the Use HTTPS is checked for the Execution Mode Load Path property, https:// is used during the execution of RAMP-TS commands.

If the Use HTTPS is checked for the RAMP Tools Mode Load Path property, https:// is used during RAMP Tools 5250 sessions.



The screenshot shows the 'RAMP-TS (Terminal Server)' configuration window. It contains the following fields and options:

- Save as deployment server:**
- IP Address:** [Empty text box]
- Port Number:** 443
- Execution Mode Load Path:** /ts/skins/ Use HTTPS
- RAMP Tools Mode Load Path:** /ts/dev/ Use HTTPS
- Private Definition Folder:** [Empty text box]

At the bottom of the window is a button labeled 'Test RAMP-TS Tools Installation and Configuration'.

Note the appropriate Port Number will depend on the port SSL is configured to use (usually 443) and whether the aXes Application server is configured to use SSL optionally or accepts SSL connections only.

See the aXes Reference - Section 8, for details about adding SSL support to aXes Application Servers.

This property is in the [Server Details](#) tab.

Use INI file

Use this property to associate a Newlook server to a Newlook ini file. The name of the file is not validated.

Leave as *DEFAULT to use Newlook.ini.

You can choose the Server – INI file combination when starting Newlook in the RAMP Tools choreographer:

At runtime the INI file is locked it to the selected server.

You can override this setting using the NLINIFILE parameter in [Web Application Start Options](#).

Use Framework Users and Authority

Use this option to enable and disable the user profile and authority management facilities that are shipped as part of the Framework.

If the user profile and authority management facilities are disabled all other user and authority related options are ignored.

This property is in the Framework [User Administration Settings](#) tab.

Use 'liteclient' license

If you check this option newlook will be started using the option to use a 'liteclient' license, otherwise newlook will be started using a default license.

This property is in the [Server Details](#) tab.

Use a Reusable Part

If this property is selected, the Framework will look for a reusable part when expanding an instance list entry at run time, otherwise it will look for a function. The reusable part must have ancestor #VF_AC023. See DF_T3507 for an example.

The component signals the addition of new instance list entries using Signal `uAddListItem`

See [Relationship Handler](#).

This property is in the Business Object [Instance List/Relations](#) tab.

Use Shared Instance List for Relationships

You should always select this option.

This property is in the Business Object [Instance List/Relations](#) tab.

Use a Webevent/WAM for Help

Select whether to use a Webevent function or a WAM to invoke the help text.

See [End-user Help \(F1\)](#).

This property is in the [Web/RAMP Details](#) tab.

Use Windows Credentials

This setting indicates that the connection to the server will be made using Windows Credentials (Kerberos / Single Signon / SSO). This means that the user's Windows profile and password is used to sign on to the server. The server must have been configured for Single Sign On, and the user enrolled, before this can be done.

See the documentation for the `CONNECT_SERVER` built-in function for more details.

If Windows credentials are used, and Framework authority is in use, (See Framework properties --> User Administration Settings --> Use Framework users and authority), the Framework will evaluate security to Framework objects based on the iSeries user profile that the user connects as, not the user's Windows profile.

This behaviour can be changed by making your own version of the IIP called Server IIP function to validate sign on. See the source for function `UF_SYSBR/UFU0005` for more details

Neither aXes nor newlook currently support Windows Credentials / Kerberos connection - so it is not possible to use Kerberos to establish RAMP 5250 sessions. You may be able to work around this by using the Server IIP function to supply an ordinary profile and password (fields `#CHK_NLUSR` `#CHK_NLPSW`) for RAMP 5250 connections, when Kerberos has been used for the initial connection. See the source for function `UF_SYSBR/UFU0005` for more details.

This property is in the [Server Details](#) tab and the [Web/RAMP Details](#) tab.

User Authorities Report File

This button produces a .csv file that contains a complete list of all the users on the system and their authority to every object on the system. The user is prompted for the name and location of the file.

This property is in the [Authorities](#) tab.

User Can Change Own Password

This option only applies to **Windows** Framework applications.

If this option is selected, users can change their own passwords by pressing the "Change Password" button on the logon screen.

In the Change Password dialog they must specify their current password and specify the new password twice. If the password change is successful, the user is returned to the logon screen with the new password already entered.

This property is in the Framework [User Administration Settings](#) tab.

User Imbedded Interface Point

Specify here the name of your user imbedded interface point file. This is an advanced property which is only required if you are using multiple Frameworks. If you have more than one Framework in the same partition, you cannot have two different versions of UF_SYSTM in the same partition. Look at the shipped source code of UF_SYSTM for details on how to create a Framework unique version.

Change this property with extreme caution. For more information, see [Imbedded Interface Points \(IIPs\)](#).

This property is in the [Framework Details](#) tab.

User Is Disabled

The administrator can use this check box to enable or disable a user profile. Disabled users are not allowed to sign on.

This property is in the [User Details](#) tab.

User Must Specify all Chosen Keys

Check this option if the end-user must specify a value for all of the selected key(s) before they can click the "Search" button.

Also see [Select the Keys of the Selected View to be used for Search Operations](#).

This is a [Program Coding Assistant](#) property.

Users May Work Offline if the Remote Server Is Not Available

This option only applies to **Windows** Framework applications.

When you select this option the user can work offline and without entering a password. The Work Offline button in the Connect dialog box will be enabled for this user.

Do not select this option if you want to validate the user profile and password of the users before allowing them access to the applications in the Framework.

Note that for users working in Design mode this option is always enabled.

This property is in the Framework [User Administration Settings](#) tab.

User Object Name/Type

Optional. This field allows you to assign a user-defined object type for this object.

Depending on your application, you may find it useful to set up object types for the objects you are working with. You can then use the object type to create generic filters or command handlers that can process different types of objects.

For example, you could write a generic filter handler that would perform processing for different kinds of business objects. You would assign the business objects User Object Types such as EMP (for employee) and CUST (for customer) and then differentiate the processing in the filter handler based on these types. You would not use the [Caption](#) property of the object for this purpose because it can be multilingual.

Similarly you could create a generic command handler that would handle both New and Copy commands. You would assign the commands using different user object types.

The User Object Type is referenced by properties `#Com_Owner.avCommandType` and `#Com_Owner.avObjectType`. See also [Framework Ancestor Components](#).

Click on the Verify Name button to have the entered value checked for uniqueness against all the other User Object Name/Type values in the Framework. Note that this check is automatically carried out when the value is edited.

Note: We strongly recommend using uppercase alphanumeric characters from the English Alphabet (ie: A->Z) only in user object types to avoid any future code page conversion issues. DBCS characters should never be used.

This property is in the [Identification](#) tab.

User Profile

Enter here the user profile which is primarily used when validating users who must sign on to use the local database.

This user profile is also used when determining the temporary directory and email address for users who do not sign on or who connect to a server.

This property is in the [User Details](#) tab.

Users Sign on Locally to Use the Framework

This option only applies to **Windows** Framework applications.

If you select this option the user will connect to the local database after signing on.

The user's profile and password are checked locally against the user profile and password specified in the User Details tab. The check is not case-sensitive.

If the user starts up the Framework using UF_ADMIN (or equivalent), the Framework will check that the user profile is an administrator. See User Details tab.

Users can be disabled if they exceed the maximum allowed signon attempts. The way in which the user is disabled can be either a simple message to the user indicating that the user profile has been disabled or it can be with a fatal error and an email to the administrator.

This property is in the Framework [User Administration Settings](#) tab.

Users Sign on to a Remote Server to Use the Framework

This option only applies to **Windows** Framework applications.

If you select this option, the user will be able to select a server and connect to it after signing on. If there is only one server defined the user will not be offered a choice of servers but will be connected automatically. If no servers are available, the user cannot sign on.

This property is in the Framework [User Administration Settings](#) tab.

Value(s) can be changed by Administrator

Specify whether the value(s) associated with this custom property can be changed by the Administrator for individual users.

This property is in the [Custom Properties](#) tab.

Warn before (days)

Number of days before password expiry to issue a warning message.

This property is in the Framework [User Administration Settings](#) tab.

***WEBEVENT Function and/or WAM Component or AJAX Routine**

Note: WEBEVENT command handlers and filters are a deprecated feature. Do not use them for new work. If you use WEBEVENT functions in your framework you will need to check the [Enable Framework for WEBEVENT Functions](#) option before you can enrol any more webevent filters or command handlers.

When snapping a filter or command handler into your Framework you must set the appropriate radio button to indicate whether you are snapping in a *WEBEVENT function, a WAM component or an AJAX routine.

Type the name of the handler or click the Find button to search for the handler. WAM components may only be used if you are using LANSAs version 10.5 or later.

You can optionally define a *WEBEVENT function and a WAM component for a filter or command handler. For example, if your Framework definition is used with a LANSAs 10.0 system and with a LANSAs 11.0 server system then you could nominate a *WEBEVENT function and a WAM component. At execution time the Framework will decide which one to use based on the capabilities of the target server system.

This property is in the [Commands Enabled](#) tab and the [Filter Snap-in Settings](#) tab.

Web Help Function/Webroutine Name

This is the name of the webevent function that is executed to present help text to end-users.

The shipped default value for webevents is UFU0002. You can replace the shipped help text presenter with your own version. Refer to the comments in shipped RDML function UFU0002 in process UF_SYSWB for the complete set of instructions on how to do this.

This property is in the [Web/RAMP Details](#) tab.

Web Help Process/WAM Name

The name of the LANSAs process that contains the function that is executed to present help text to end-users or the WAM which is used to invoke the help text. The shipped default value is UF_SYSWB. You can replace the shipped help text presenter with your own version. Refer to the comments in shipped RDML function UFU0002 in process UF_SYSWB for the complete set of instructions on how to do this.

Alternatively you can use a WAM to display the help text. The shipped UF_SY002 offers a basic example of how to use a WAM to display application help.

This property is in the [Web/RAMP Details](#) tab.

Web Help Window Features

This property allows Framework designers to specify the properties of the Help window which is opened when an end-user presses F1.

The value specified here is passed as a parameter to the JavaScript window.open method when the help window is opened. As such it must be validly formatted as a window.open parameter or unexpected results or application failures may occur.

Refer to Microsoft JavaScript documentation for more details about this parameter.

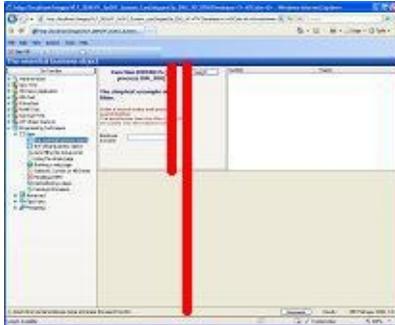
The shipped default value is

`width=600,height=425,directories=no,toolbar=no,menubar=no,scrollbars=yes,rel`

This property is in the [Web/RAMP Details](#) tab.

Web Initial Filter Pane height (%)

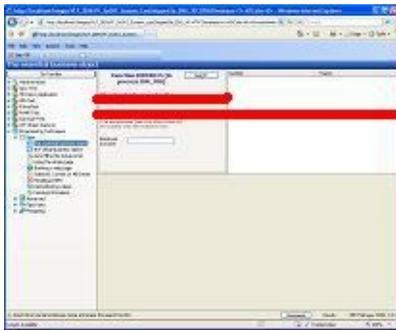
This property gives the designer some control over the height of the filter and instance list panes, relative to the Framework, when the web Framework starts up. The percentage is the ratio of the height of the filter/instance list panel to the height of the Framework:



This property is in the [Web/RAMP Details](#) tab.

Web Initial Filter Pane width (% of right panel)

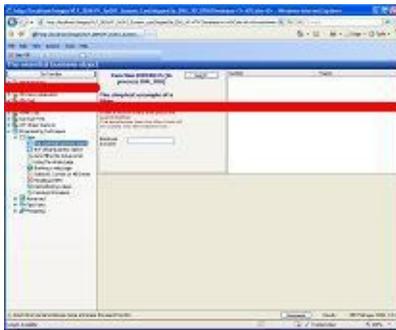
This property gives the designer some control over the width of the filter pane relative to the right panel when the web Framework starts up. The percentage is the ratio of the width of the filter to the width of the main right hand panel:



This property is in the [Web/RAMP Details](#) tab.

Web Initial Navigation Pane width (%)

This property gives the designer some control over the width of the navigation pane when the web Framework starts up. The percentage is the ratio of the width of the navigation pane to the width of the Framework:



This property is in the [Web/RAMP Details](#) tab.

Web Server Caption

Enter here a short description that that you can use to recognize the server.

For example, you may be using two web servers while developing applications. One might be the web server running on your desktop PC and the other your main System i web server. In this case you need need to establish different configuration details for the servers.

This property is in the [Developer Preferences – Web Server](#) tab. It is also a [Web Configuration Assistant](#) property.

Web Server IP Address

This is the IP address of the webserver (without http:// or the port). You may also use a name that resolves to the ip address.

If you are using a webserver that is on your development PC, you might use localhost, or 127.0.0.1, or the PC name as your webserver IP address.

This is a [Web Configuration Assistant](#) property.

Web Server Port

This is the port used for http requests to the web server. It is usually 80. If you don't want a port to be specified, use value 0.

This is a [Web Configuration Assistant](#) property.

Width

Specify the width of the image/web page, in pixels, to be displayed on the Sign On form.

This property is in the Framework [User Administration Settings](#) tab.

View as Drop Down on Toolbar

Select this option to display the [Application Bar](#) as a drop-down list on the toolbar.

This property is in the [Framework Details](#) tab.

View as A Single Tree

Select this option to display the [Application Bar](#) as a tree view.

This property is in the [Framework Details](#) tab.

View as Two Lists Side By Side

Select this option to display the [Application Bar](#) as two lists side by side.

This property is in the [Framework Details](#) tab.

View as Two Lists Over and Under

Select this option to display the [Application Bar](#) as two lists above and below.

This property is in the [Framework Details](#) tab.

View Used to Read the Child Data

Select the view of the physical file that contains the data for the child object.

This is a [Program Coding Assistant](#) property.

Windows Code Page

This is the Microsoft Windows code page identifier you normally use with this language.

Typically the default values are correct. Refer to your product vendor and/or Microsoft Windows documentation before changing this value.

This property is in the [Web/RAMP Details](#) tab.

Visual Identifiers for Building VisualID1 and VisualID2 Values

Specify the visual identification protocol you wish to use for this business object.

For example, for a "Products" business object you might decide to specify fields PRODNO (Product Number) and PRODDESC (Product Description) as the visual identifiers that are typically used to identify a product to end-users of your application.

Any field name can be specified in this list (even ones not currently defined in the LANSAs data dictionary). The fields specified do not necessarily have to be columns in any physical file that you specify at the top of the form.

See [Visual Identifiers](#) and [Programmatic Identifiers](#) for more details of business object identification protocols.

This is a [Program Coding Assistant](#) property.

Visual Style Base Style

This style sets the appearance for the basic look of the object.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

Visual Style Protected Fields and Areas

This style sets the appearance for protected areas.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

Visual Style Dark Background Small Font

This style sets the appearance of parts of the interface with small fonts and dark background.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

Visual Style Dark Background Large Font

This style sets the appearance of parts of the interface with large fonts and dark background.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

Visual Style Status Bar Fields

This style sets the appearance of status bar fields.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

Visual Style URLs

This style sets the appearance of URLs.

This option only applies to **Windows** Framework applications.

This property is in the [Visual Styles](#) tab.

VLF.NET Screen Layout Persistence Level

Specifies the way that VLF.NET application screen layout arrangements are to be persisted and managed. The default Framework option indicates that a single screen layout applies to the entire framework.

The Business Object option indicates that each business object may have its own screen layout.

This property is in the [Framework Details](#) tab.

Your Framework Version Number

You can specify your own Framework version number. It consists of four numbers, each of which must be in the range 1 to 9999999.

The version number is specified in the Primary Version . Secondary Version . Build Number . Revision Number format.

The version number is significant when using VLF.NET as it indicates to Internet Explorer that the version of your Framework has been updated on the web server.

See [Automatically Increment when Saving](#) and [Show in Help About Text](#).

This property is in the framework's [Identification](#) tab.

Appendix

What is Included in the Framework?

What is Included in the Framework?

LANSA Repository Objects:

- Demonstration Repository Objects
- User Repository Objects
- Framework Repository Objects

Operating System Objects

- Permanent File Objects
- Temporary or Semi-Persistent Files

Things You Should Back Up

LANSA Repository Objects

Demonstration Repository Objects

User Repository Objects

Framework Repository Objects

Demonstration Repository Objects

Name	Type	Description	Deploy to Execution Repositories(Windows)	Deploy Execut Reposi
DF*	Function	Demonstration Functions	Not usually	Not Us
DF_*	Reusable Part	Demonstration application components	Not usually	Not us
DF_*	Form	Demonstration application forms	Not usually	Not us
DF_*	ActiveX	Demonstration application activeX components	Not usually	Not us
DF_EL*	Simple Field	Fields are used by demonstration components	Not usually	Not us
DF_EL*	Web components	as above	Not usually	Not us
DF_EL*	HTML pages	as above	Not usually	Not us
DF_MAXNUM	Field			
DF_PROC	Process	Demonstration Process	Not usually	Not us
DF_VS*	Visual Style	Visual styles are used by the demonstration components	Not usually	Not us

DFXPROC	Process	Framework demonstration RDMLX process	Not usually	Not us
DM*	WAM	Demonstration WAMs	Not usually	Not us
DM_*	Reusable part	Demonstration reusable parts	Not usually	Not us
DW*	Function	Demonstration Functions (web)	Not Usually	Not U:
DW*	Web components	Used by web demonstration functions	Not Usually	Not U:
DW*	HTML pages	referred to by the DW* web components	Not Usually	Not U:
DW_*	Process	Demonstration Processes (web)	Not usually	Not us
MTXTDF_*	Multilingual Variable	The multilingual variables are used by demonstration components	Not usually	Not us

Notes:

1. Modifications will be overwritten when upgrading Framework versions.

User Repository Objects

Name	Type	Description	Deploy to Execution Repositories(Windows)	Deploy to Execution Repositories
MTXTUB	Multilingual variable	Multilingual variables	Not usually	Not usually
MTXTUF_	Multilingual Variable	Multilingual variables	Not usually	Not usually
UB*	Weblet	User Button Weblets	No	Yes
UB_*	Fields	Buttons for web functions	No	Yes
UB_*	web components	Buttons for web functions	No	Yes
UB_*	html pages	Buttons for web functions	No	Yes
UF_*	Field	Fields	No	No
UF_*	Reusable part	Reusable parts	Yes	No
UF_EXEC UF_ADMIN UF_DEVEL UF_DESGN	Form	Framework entry points for normal end-user, administrator end-user, Framework Developer and Framework designer.	Usually	No
UF_IB001	Reusable Part	Icon and Bitmap	Usually	No

		Loader		
UF_IM*	Bitmaps	Start up and demonstration bitmaps	Not usually	No
UF_SYSBR Functions UFU*	Process with functions starting with UFU*	User IIP (includes shipped code table data handlers)	Yes	Yes
UF_SYSTM	Reusable Part	User Imbedded Interface Points (IIPs)	Usually	No
UF_SYSWB Functions UFU*	Process with functions starting with UFU*	User IIP (F1 Help display for web)	No	Yes
UF_VS*	Visual Styles	Visual styles used in the Framework	No	No

Notes:

1. Modifications will be overwritten when upgrading Framework versions. To avoid this problem refer to instructions in the shipped source code.

Framework Repository Objects

Name	Type	Description	Deploy to Execution Repositories(Windows)	Deploy Execut Reposi
MTXTFP_	Multilingual variable	Weekdays etc.	Not usually	Not us
FPTAB	Database file	Code Table data	Yes	Yes
MTXTVF_*	Multilingual Variable	Multilingual variables	Not usually	Not us
VF*	Form	Framework forms	Yes	No
VF*	Function	Framework functions	No	Yes
VF_*	Field	Framework fields	Yes	Yes
VF_*	Web component	Framework web components	No	Yes
VF_AC*	Reusable Part	Framework ancestor classes	Usually	No
VF_AW*	Reusable part	WAM ancestor classes	No	Yes
VF_AX*	ActiveX	Framework ActiveX classes	Usually	No
VF_BM*	Bitmaps	Shipped Bitmaps (if any)	No	No
VF_CH*	Reusable Part	Default command handlers	Not usually	No
VF_CO*	Reusable	Communications	Usually	No

	Part	and connections		
VF_EL*	Simple Field	Fields used by Framework	Not Usually	Yes
VF_EL*	web components	web components used by the Framework	No	Yes
VF_EL*	html pages	html pages used by the Framework	No	Yes
VF_ER	Reusable Part	Error Handler	Usually	No
VF_FL*	Reusable Part	Default filters	Not usually	No
VF_FP*	Reusable Part	Persists Object (XML defined)	Usually	No
VF_GP*	Reusable Part	General Purpose Objects	Usually	No
VF_HE*	Reusable Part	Help and Assistant	Usually	No
VF_IB*	Reusable Part	Icon and Bitmap Library Objects	Usually	No
VF_IC*	Icons	Shipped Icons (if any)	No	No
VF_IM*	Images	Shipped Images (if any)	No	No
VF_LM*	Reusable Part	Instance List Management	Usually	No
VF_LW*	Reusable part	Instance List Management	No	Yes
VF_PC*	Reusable Part	Virtual ClipBoard	Usually	No

		Manager		
VF_PR001	Process with with functions VFU*	Framework internal functions	Yes	Yes
VF_PR*	Processes except for VF_PR001 with functions VFU*	Framework internal functions	No	Yes
VF_SK	Reusable Part	Skeleton Layout	Not Usually	No
VF_SW*	Reusable part	WAM Framework Services Manager	No	Yes
VF_SY*	Reusable Part	Framework Managers	Usually	No
VF_UM*	Reusable Part	User Interface Management	Usually	No
VF_VS*	Visual Style	Visual Styles	Not Usually	Not Us
VFPPF04	Database file	Used for record format only	No	No
VFPPF06	Database file	User data	Yes	Yes
VFPPF07	Database file	User authority data and other custom properties	Yes	Yes
VFU*	WAM	Commands and Filters	No	No

vlf_*

Weblet

Framework
WAM layout
weblet

No

Yes

Notes:

1. Behavioral modifications are achieved by redefining ancestor methods.
2. Behavioral modifications are achieved by via imbedded interface points (IIPs).

Operating System Objects

Permanent File Objects

Temporary or Semi-Persistent Files

Permanent File Objects

File NameFormat	Resides In	Descript
DF*.dll	x_ppp\execute	Executa demons compon
DF*.gif	x_ppp\execute	Executa demons compon
DF_DET18*.htm	x_ppp\execute	Executa demons compon
DF_DET18*.xsl	x_ppp\execute	Executa demons compon
dm_*.dll	x_ppp\execute	Executa demons compon
DW*.dll	x_ppp\execute	web exe demons compon
DW*.s	x_ppp\execute	web ske compile
DW_T12_DWT1202_AboutLANSA.htm	x_ppp\execute	Demons applicat
DW_T12_DWT1203_AboutLANSA.doc	x_ppp\execute	Demons applicat
DW_T12_DWT1204_Table*.XLS	x_ppp\execute	Demons applicat
DW_T12_DWT1205_VLFOview.ppt	x_ppp\execute	Demons applicat

DW_T12_DWT1206_AboutLANSA.pdf	x_ppp\execute	Demons applicat
ENG_vf* / FRA_vf* / JPN_vf* etc .htm	x_ppp\execute	Program Assistar
ENG_vf* / FRA_vf* / JPN_vf* etc .vfi	x_ppp\execute	Program Assistar
ENG_vf* / FRA_vf* / JPN_vf* etc .vft	x_ppp\execute	Program Assistar
FP_TABLE_*.dat	x_ppp\execute	Data for tables
FPTAB.dll	x_ppp\execute	io modu FPTAB
LANSA047.chm	x_lansa\execute	Framework Check I
LANSA048.chm	x_lansa\execute	Framework and Tut
Lansa049.chm	x_lansa\execute	RAMP
NL_Load_Test.htm	x_ppp\execute	RAMP
U_BIF985.DLL	x_lansa\execute	User de function
U_BIF987.DLL	x_lansa\execute	User de function
UF*.css	x_ppp\execute	Style sh
UF_*.dll	x_ppp\execute	Executa Framework
UF_IM*.gif	x_ppp\execute	Start Up images
UF_JS001.JS	x_ppp\execute	Scripts
VF_* (except VF_IC*).gif	x_ppp\execute	Program Assistar

VF_*.dll	x_ppp\execute	Executa Framew
VF_Framework_Virtual_Clipboard_Default.dat	x_ppp\execute	Default shipmen settings Framew virtual c
VF_HE003.dat	x_ppp\execute	Help top referenc
VF_HT*.htm	images directory and also x_ppp\execute	web Fra
VF_ICnnn.gif	images directory and also x_ppp\execute	Other Ir Framew Images
VF_ICnnn1.gif	images directory and also x_ppp\execute	16x16 p web Fra
VF_ICnnn3.gif	images directory and also x_ppp\execute	32x32 p web Fra
VF_MACRO.sid	x_ppp\execute	newlook RAMP
vf_multi*.js	x_ppp\execute	Multilin used in the Visu Framew
VF_SY001_Nodes YYYYMMDD_HHMMSS.XML	x_ppp\execute	RAMP : definitic

VF_SY001_Nodes.XML	x_ppp\execute	RAMP : definitic
VF_Sy001_Servers.xml	x_ppp\execute	Server I
VF_Sy001_Servers_YYYYMMDD_HHMMSS.xml	x_ppp\execute	Server I version
VF_Sy001_System.xml	x_ppp\execute	Framew
vf_sy001_system_*.JS	X_ppp\execute	The equ system.: script sc
VF_Sy001_System_*_Launch_*.htm	x_ppp\execute	Initiates Framew by Fram
VF_SY001_System_*_BASE.HTM	x_ppp\execute	Initiates Framew by Fram Contain VF_sy0
vf_sy001_system_choice.txt	x_ppp\execute	Framew selectio
vf_sy001_system_Preferences_username.xml	x_ppp\execute	System
VF_Sy001_System_YYYYMMDD_HHMMSS.xml	x_ppp\execute	Framew saved ve
VF_Sy001_Tables.xml	x_ppp\execute	Code Ta
VF_Sy001_Users.xml	x_ppp\execute	User De
VF_Sy001_Users_YYYYMMDD_HHMMSS.xml	x_ppp\execute	User De version
vf_sy004_*.dat	x_ppp\execute	HTML d
vf_sy120.htm	x_ppp\execute	newlook

		manage
vf_sy120.js	x_ppp\execute	newlool manage
UF_SY120.JS	x_ppp\execute	File to s JavaScr
VF_SY131_Filter_ENG.HTM	x_ppp\execute	Create I objects : Prototy
VF_SY131_Handler_ENG.HTM	x_ppp\execute	Create I objects : Prototy
VF_UM*.GIF	x_ppp\execute	Demons gifs
vf_um703.htm	x_ppp\execute	newlool
vf_um703.js	x_ppp\execute	newlool
VF_UM835.xml	x_ppp\execute	Generic
VF_User_Virtual_Clipboard_Default.dat	x_ppp\execute	Default shipmen settings virtual c
VF_VS*.css	images	style sh Framew
VF_XP*.nlg	x_ppp\execute	Framew macros
VF_XXnnn.js	images	web Fra javascri
VFPP*.dll	x_ppp\execute	io modu files (us storage)

Notes:

1. X_ppp where "ppp" is the LANSAs partition identifier
2. Special considerations apply to deploying these objects to multi-Framework environments.
3. Or equivalent. These file(s) may have different names in differently configured Frameworks.
4. These files need to be deployed if your Framework has elected to make use of them. You can deploy these Framework DLL's by using the VL Deployment tool. Use the option to add Non-LANSAs Objects under the Other Objects Tab sheet. You will have to create a new path name e.g. SYSPATH that points to the System Execute folder to install the DLL's into the correct folder on the target PC.
5. These files may be altered by administrators at deployed sites.
6. These files are used by shipped Frameworks, but are normally replaced with user defined versions before shipment.
7. For Windows execution environments VF_SY001_SYSTEM.XML is normally deployed. For web execution environments VF_SY001_System.xml is not normally deployed because it is imbedded in VF_SY001_System_*_BASE.HTM. VF_SY001_System_*_BASE.HTM is deployed for web execution environments.
- 11.Required for RAMP.
- 13.Windows Web server only

Temporary or Semi-Persistent Files

Name Format	Resides In	Description	Deploy to Execution Environments	SeeNotes
ppp_User_ _Virtual_ClipBoard.Dat	Directory defined in system variable *Temp_Dir	User virtual clipboard data storage	Not Usually	1
ppp_Framework_ _Virtual_ClipBoard.Dat	Directory defined in system variable *Temp_Dir	Framework virtual clipboard data storage	Not Usually	1
Bbbbbbbbbbbb_Part_1.Lst Bbbbbbbbbbbb_Part_2.Lst	Temporary directory defined for user. Defaults to system variable *Temp_Dir	Saved business object instance list details.	Not Usually	2

Notes:

1. Where "ppp" is the LANSAPART partition identifier
2. Where "bbbbbbbbbbbb" is the unique Internal Identifier of the associated business object.

Things You Should Back Up

This is a list of the things that you should back up from your Framework:

Object	When	SeeNotes
Your LANSAs Repository	After any changes	
Framework_Export*.zip	As required	
VF_SY001_System*.XML	Save after Framework design sessions	1
VF_SY001_Users*.XML	Save after editing user details	1
VF_SY001_Servers*.XML	Save after server sessions	1
VF_SY001_Tables*.XML	Save after editing code tables	1
VF_SY001_Nodes*.XML	Save after editing RAMP	1
VF_SY120.JS	Save after editing RAMP	
UF_* (User repository objects)	After any changes to these objects	

Notes:

1. Or equivalent. These files may have different names in differently configured Frameworks.
2. Depending upon the Framework configuration these files will reside either in the LANSAs for the Web images directory or a subdirectory of it named ppp_FLA (where ppp is the identifier of the partition being used).