http://www.lansa.com

Newsletter
January 2005

LANSA Launch 2005

The LANSA Launch 2005 User Meeting for our Customers and Partners will take place on ***March 15 in Antwerp*** Belgium.

The guest speaker will be ***Mark Duignan***, Manager of the Application and Integration group, LANSA Product Center.

Register online to reserve your place.

The Seven Best Reasons to Step Up to LANSA 2005

1 **A New Way to Design and Build Web Browser Applications.**
Using a new LANSA feature called WAMs (Web Application Modules), you can substantially reduce the learning curve and cost of developing Web browser applications. Developers with 5250 green screen backgrounds can now quickly and easily learn how to produce the sophisticated Web browser-based applications. A complete WYSIWYG form painter is included.

2 **A Better Way to Build or Extend any Type of New or Existing LANSA Application (even 5250 Green Screen Ones).**
The LANSA development workbench (IDE) has been completely replaced in LANSA 2005. Producing 5250, Web, or Windows application is now easier and faster. Developers using this workbench are demonstrably more productive than those using 5250 screens for development.

In This Issue

3 **A New Application Framework to Rapidly Prototype, Develop and Implement Windows and/or Web Browser Applications.**
Developers with 5250 green screen applications can now easily and quickly produce sophisticated Windows and/or Web applications. The learning curve is short and your development costs are substantially reduced. If you are contemplating 5250 application modernization by using Windows or Web browser user interfaces to your old 5250 – this is the way to go.

4 **An Upgraded Version of LANSA Integrator.**
The range of integration services offered has been significantly expanded to include support for SOAP, EDI, SMS, and DSV and for Zip and PDF file creation. Benefiting from relaxed language limitations too, LANSA Integrator empowers developers to use Java Services seamlessly in their applications, including vastly simplified access to and implementation of Web Services.

5 **A New Set of Built-in Functions.**
General LANSA programming tasks will be even easier and simpler than before. The areas of improvement include string handling and formatting, encryption and decryption, data conversions and reading/writing to flat files (e.g., on the iSeries IFS).

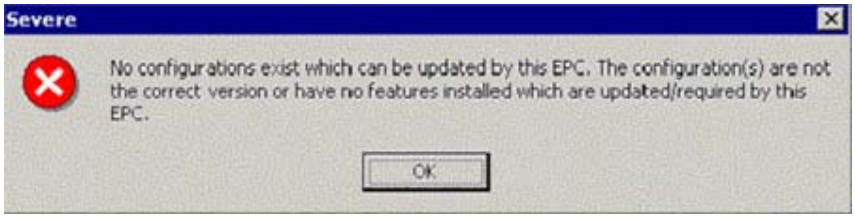
6 **Removal of Most Limitations Imposed by RPG on LANSA Applications.**
Most V10.0 limits on field lengths, field types, browse and working list size limits, I/O field limits, etc, have been removed in LANSA 2005. Variable field lengths, very long fields (up to 64k), integer and float numeric fields and BLOB/CLOB (binary and character large object fields) are all now supported.

7 **An Enhanced Syntax for RDML Programming Language.**
Write less code with improved string handling. The enhanced syntax is more familiar and comfortable to brand new LANSA developers. This reduces both your development and training costs.

Cannot apply any EPCs to a V10.0 Visual LANSA configuration in a Client/Server development environment

Description

The following error can be generated when trying to apply an EPC to a Visual LANSA V10.0 environment or when installing a new product/feature to an existing V10.0 configuration. For example, installing the LANSA for the Web Utilities to an existing configuration.



This problem is caused by corruption of the liiconfig.txt file on the Network Server. The 2 currently known causes of liiconfig.txt corruption are:

1. When performing the Network Client install, the user incorrectly selects the server directory as the local installation directory. This causes the liiconfig.txt on the server to be converted from a Network Server version to a Network Client version.
2. After successful network server/network clients installs, user selects to install another LANSA feature on the network client ie. Web Administrator, Web utilities etc. Installing the new LANSA feature to an existing network client configuration also corrupts the liiconfig.txt on the server.

Solution

Note: As this problem is in the Visual LANSA V10.0 installation, this issue cannot be fixed by EPC for V10.0. However, there are currently two ways to work around it in 10.0:

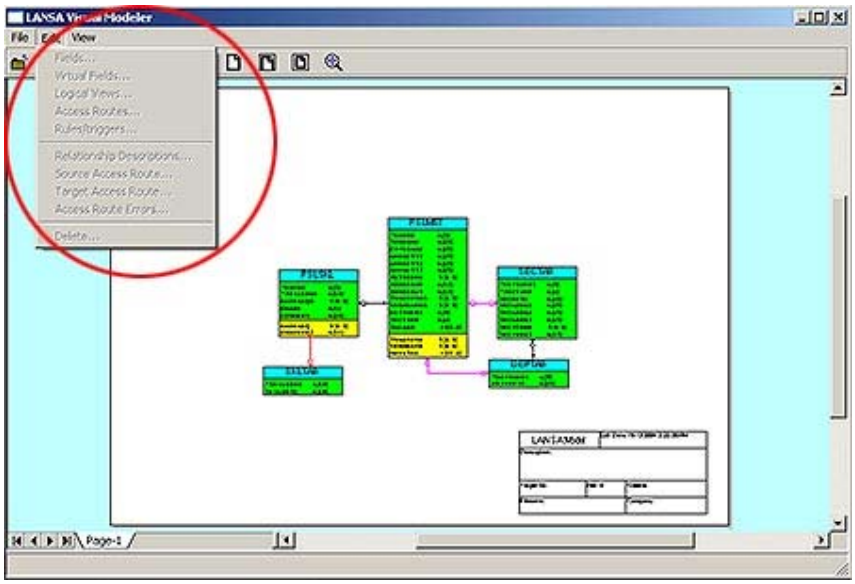
1. Reinstall Visual LANSA on the network server selecting the same installation type and including all the previous installed features ie. Web support. Note that all EPCs will need to be re-applied.
2. Contact your local LANSA distributor and inform them that you have experienced this problem. They will request specific information which will allow LANSA support to rectify the network server liiconfig.txt.

We have put corrections in V11.0 to stop the known causes above from occurring.

Edit menu options greyed out in Visual LANSA V10.0 Physical Modeler

Description

The Visual LANSA Modeler in V10.0 comprises of the Physical Modeler and the Logical Modeler. For further information regarding Modeling in Visual LANSA, refer to the Visual LANSA Logical Modeler Guide in the online documentation. The Physical Modeler should be automatically available after a successful Visual LANSA V10.0 install. However, it has occurred that when you open the Physical modeler, the Edit menu is greyed out.

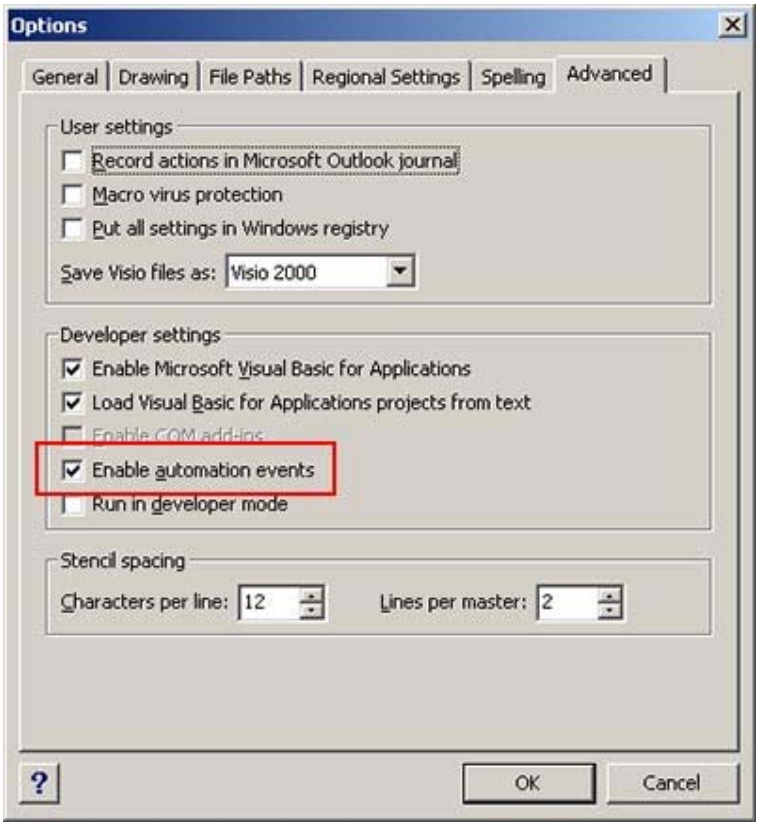


Solution

The simplest possibility to check is whether the Edit menu options are greyed out because there is no file selected. If you use the mouse to select any of the files, the Edit menu options will become available. Check this as the first possibility ie. select a file/give it focus and then check the Edit menu again.

If the Edit menu options are still greyed out, try the following possibility:

- Locate the Visio document LANSAxxx.vsd in the LANSA folder (where 'xxx' is the development language being used)
- Open this document in Visio
- Select Tools -> Options -> Advanced
- Ensure 'Enable Automation Events' is checked.
- Save the document


Options
General
Drawing
File Paths
Regional Settings
Spelling
Advanced
User settings
Record actions in Microsoft Outlook journal
Macro virus protection
Put all settings in Windows registry
Save Visio files as:
Visio 2000
Developer settings
Enable Microsoft Visual Basic for Applications
Load Visual Basic for Applications projects from text
Enable COM add-ins
Enable automation events
Run in developer mode
Stencil spacing
Characters per line:
12
Lines per master:
2
OK
Cancel

How to Confirm JAVA Version on iSeries

Description

There may be instances where the latest installed version of Java has actually not been activated on the iSeries and which can cause problems trying to run LANSA Integrator.
To check to see which version the iSeries has currently activated run the command:
JAVA *VERSION
which will return the current Java version being used ie:
Version JVM V5R2M0 JDK 1.3.1
Currently, the lowest version that should be used (and installed) is JDK 1.4.

There are three methods to allow the correct Java version.

1. When calling STRJSM, specify the Java version.

```
Start Java Service Manager (STRJSM)

Type choices, press Enter.

Instance . . . . . .   *DEFAULT
JVM Version  . . . . . .   *DEFAULT      *DEFAULT, *JVM12,
*JVM13, *JVM14, *JVM15
Option . . . . . .   *NONE         *NONE, *VERBOSE,
*VERBOSEGC..
Garbage collect initial size . .   40000         256-139264000
kilobytes
Time zone  . . . . . .   *DEFAULT
```

2. For LANSA Integrator, edit the SystemDefault.properties file in the system folder to point to the correct version of JAVA.

   /jsm/instance/system/SystemDefault.properties

   Edit the file with the following line:

   java.version=1.3' or whichever version you require.

   e.g:

   #!<studio-project id="20040000-000000" name="test">
   #
   java.version=1.4
   #
   #!</studio-project>

   Note: You will need to restart JSM after making this change.

3. Or you can change the overall JAVA version on the iSeries:

- Use command WRKLNK
- Use option 2 to the directory /QIBM/UserData/Java400
- Edit the file ‘/QIBM/UserData/Java400/SystemDefault.properties’ with the following line:
  'java.version=1.3' or whichever version you require.

How to Call a LANSA Function from a 3GL

Description

How to call a LANSA function from a C program (3GL) in LANSA for Windows and LANSA for iSeries.

Answer

Depending on the platform of execution, there are 2 ways of calling a LANSA function from a 3GL:
Calling a LANSA function from a 3GL on iSeries and
Calling a LANSA function from a 3GL on Windows.

On iSeries

1. You will need to have a refer to the Online documentation under 'LANSA Open Systems Utility Guide' (OSU) regarding calling LANSA processes and functions from CL or RPG. There is a section titled 'Direct Calling of LANSA Functions' that has sample source codes in RPG that can achieve what you want. You can still use the RPG program, however as stated below, you will need to call the RPG program via your C program.

   Also note that the basic information in the sample source code applies equally to calling from C programs.

2. Using the sample program UD@FUNC1 as a basis, create an RPG program (say MYCALL) that calls the required LANSA function. There are many variations on this (see following details headed 'Variations').

3. Call MYCALL from the C program. To do this you need to use the normal C compiler options that indicate that you want to call an RPG program.

For example:
You need standard C pragmas:
#pragma map(MyProgramCall, "MYCALL")
#pragma linkage(MyProgramCall, OS, nowiden)

You need a prototype:
void MyProgramCall (void);

You need to make the call in you code:
MyProgramCall();

Notes: See the documentation for what to change in order to call a particular function and also how to pass parameters.

Variations:

1. If you have many different functions to call then change MYCALL to receive the name of the LANSA function to be called as a parameter from the C program as a char 7 value:

   You need a different C prototype:

   void MyProgramCall (char *);

   You need to make the call passing the 7 character function name (actually 8 in length with the terminator):

   MyProgramCall("F001 ");

2. If you need to make lots of calls back and forwards change MYCALL to be reentrant (i.e.: do not set on LR) and avoid repeating the relatively expensive calls UD@CALL1 and UD@CALL2 every time MYCALL is invoked. You would only need to make these calls the first time MYCALL is invoked.

3. Passing data structures, lists and using the exchange list are all possible. Refer to the documentation in the OSU guide for the exact details.

On Windows

You need to use the system() api or CreateProcess() api to call X_RUN with appropriate parameters as documented in LANSA documentation.

User parameters can be passed in a file (TRANSFORM_FILE BIF) or through registry (GET_REGISTRY_VALUE BIF).

For information purposes: Version 11.0 has a new parameter - UDEF - so it can be passed on the command line. This can be queried from RDML using GET_SESSION_VALUE.

Assigning Dynamic Button Images

Description

Method for dynamically assigning images to buttons.

To do this, create a non-visual reusable part, which will be referred to as the image librarian. In this reusable part, define all of the available images in the repository in a collection of bitmap objects keyed by their names:

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_OBJT)

* Collection of Bitmaps - the 'library'
DEFINE_COM CLASS(#prim_kcol<#Prim_bmp #Std_Obj>) NAME(#Bitmaps)

* Property to retrieve a bitmap
Define_Pty Bitmap Get(Get_Bitmap)

PtyRoutine Get_Bitmap
Define_Map *output #Prim_bmp #Pty_002 pass(*By_reference)
Define_Map *input #Std_Obj #Pty_001

* Set the Reference
Set_Ref #Pty_002 #Bitmaps<#Pty_001.Value>
Endroutine

EVTROUTINE HANDLING(#COM_OWNER.CreateInstance)
OPTIONS(*NOCLEARMESSAGES *NOCLEARERRORS)
* Hardcode the images in here
Set_ref #Bitmaps<'VB_OPEN'> #VB_OPEN
Set_ref #Bitmaps<'VB_CLOSE'> #VB_CLOSE
Set_ref #Bitmaps<'VB_CD'> #VB_CD
* Etc...
ENDROUTINE

END_COM
```

This creates a library with 3 bitmaps. The property routine uses the standard input/output parameters to pass back a reference to the bitmap when given a string input.

Then in the button reusable part, include this librarian with a define_com; allowing access to the images. *Note that the scope parameter has been used to define this object as a shared component. This means that many different forms and reusable parts can reference this object but only one instance of the object is loaded into memory*:

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_SPBN) DISPLAYPOSITION(1) HEIGHT(18)
LEFT(0) TABPOSITION(1) TOP(0) WIDTH(58)
* Define an Image Library in this Component - note the use of SCOPE
DEFINE_COM CLASS(#ImgLib) NAME(#ImgLib) SCOPE(*SHARED)

define_pty name(uImage) set(SetuImage)

ptyroutine name(SetuImage)
define_map for(*input) class(#std_obj) name(#InputName)

* Now retrieve the image from the image library and assign it to the button
set #com_owner image(#ImgLib.Bitmap<#InputName>)
endroutine

END_COM
```

Note: This method is not as obvious and technically not completely dynamic.
This same technique is used in the Visual LANSA Framework when assigning images to buttons.

V5R2 Client Access Users and the LANSA Host Monitor

Description

If you have V5R2 Client Access, you may have problems trying to start new 5250 sessions while the LANSA Host Monitor is running. To rectify this problem you would need to apply Client Access service pack SI14294.

Refer to either of these URLs for more information:
http://www-912.ibm.com/a_dir/as4ptf.nsf/0/889ef02beb67f66286256f4f00528a6f?OpenDocument
http://www-912.ibm.com/eserver/support/fixes/DisplayCoverLetter.jsp?enableOrder=Y&fixid=SI16136

JIT may not install packages in order expected

Description

When using the Visual LANSA Deployment Tool, the Just In Time (JIT) package installer is accesses the packages in alphabetical order.

Note, in Windows XP SP2, file sorting has changed so embedded numbers are now recognized. i.e For packages named PK1, PK10 and PK2, packages will now install in the correct order of PK1, PK2 and PK10 whereas previously the install order would have been PK1, PK10 and PK2.

However, this issue is discussed in the documentation for 10.7, under Package Dependencies (Chapter 4.8 of the LANSA Application Deployment Tool documentation). It mentions *"in some cases, you may deploy your application using more than one package. Naturally, it is important for these packages to be installed in the correct sequence. To ensure this, you would use Package Pre and Co-Requisites."*

Solution

Each package that is deployed, should have a prerequisite of the package before it. The package installer will loop through the packages, skipping those that fail the pre-requisites, and coming back to install them later.

FormStyle property of a Form

Use this property to specify the behaviour of the form at run-time.

The property values are:

- **Normal**: The form is a standard form.
- **Owned**: When this form is made a member form of another form, it will be closed when the owner form is closed.
- **StayOnTop**: The window will stay on top of any other windows open on the desktop.

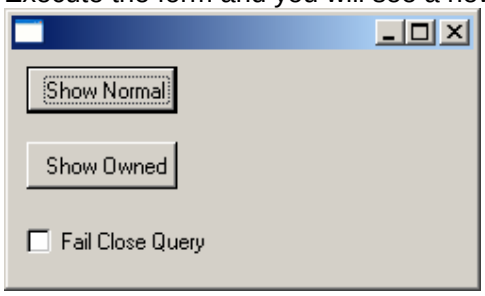
When you set the style of a form to **NormalChild**, **OwnedChild** and **StayOnTopChild**, the form will be hidden or minimized when its parent form is hidden or minimized and it will be automatically closed when its parent form is closed. An OwnedChild form will stay on top of its owner form and it does not appear on the task bar.

To see the difference between NormalChild and OwnedChild forms, copy and paste the following code to a form called FORMOWNEX, compile and execute it.

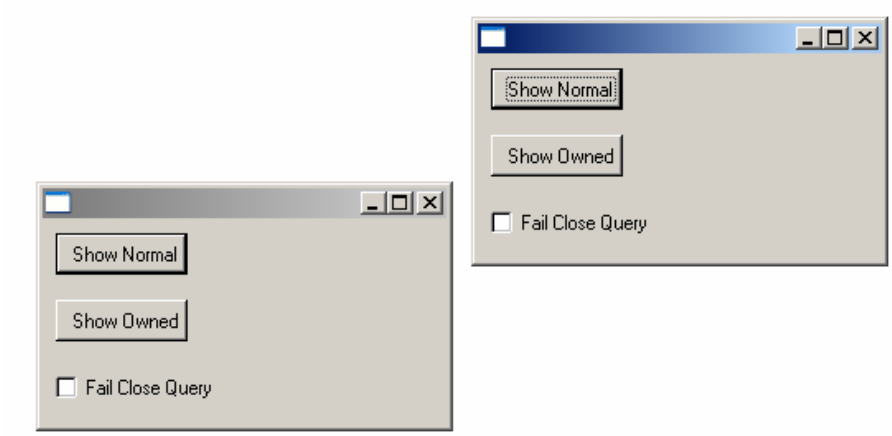
```
FUNCTION options(*DIRECT)

BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(150) LEFT(296) TOP(111) WIDTH(253)
DEFINE_COM class(#PRIM_PHBN) name(#PHBN_1) CAPTION('Show Normal')
DISPLAYPOSITION(1) LEFT(8) PARENT(#COM_OWNER) TABPOSITION(1) TOP(8)
DEFINE_COM class(#PRIM_PHBN) name(#PHBN_2) CAPTION('Show Owned')
DISPLAYPOSITION(2) LEFT(8) PARENT(#COM_OWNER) TABPOSITION(2) TOP(48)
DEFINE_COM class(#PRIM_CKBX) name(#CKBX_1) CAPTION('Fail Close Query')
DISPLAYPOSITION(3) LEFT(8) PARENT(#COM_OWNER) TABPOSITION(3) TOP(88)
DEFINE_COM class(#FORMOWNEX) name(#NORMALFORM) FORMSTYLE(NormalChild)
DEFINE_COM class(#FORMOWNEX) name(#OWNEDFORM) FORMSTYLE(OwnedChild)

EVTROUTINE handling(#PHBN_1.Click)
IF_REF com(#Com_Owner.FormOwner) is(*Null)
CHANGE field(#STD_NUM) to('#Com_Owner.Left + #Com_Owner.Width + 10')
ELSE
CHANGE field(#STD_NUM) to('#Com_Owner.Left + 10')
ENDIF
SET com(#NormalForm) FORMOWNER(#Com_Owner) LEFT(#Std_Num)
INVOKE method(#NormalForm.ShowForm)
ENDROUTINE

EVTROUTINE handling(#PHBN_2.Click)
IF_REF com(#Com_Owner.FormOwner) is(*Null)
CHANGE field(#STD_NUM) to('#Com_Owner.Top + #Com_Owner.Height + 10')
SET com(#OwnedForm) TOP(#Std_Num)
CHANGE field(#STD_NUM) to('#Com_Owner.Left + #Com_Owner.Width + 10')
ELSE
CHANGE field(#STD_NUM) to('#Com_Owner.Left + 10')
ENDIF
SET com(#OwnedForm) FORMOWNER(#Com_Owner) LEFT(#Std_Num)
INVOKE method(#OwnedForm.ShowForm)
ENDROUTINE

EVTROUTINE handling(#COM_OWNER.CloseQuery) options(*NOCLEARMESSAGES
*NOCLEARERRORS) CONTINUE(#Option)
IF cond('#CKBX_1.ButtonState = Checked')
SET com(#Option) VALUE(False)
```

ENDIF
ENDROUTINE
END_COM

Execute the form and you will see a new task on your taskbar:

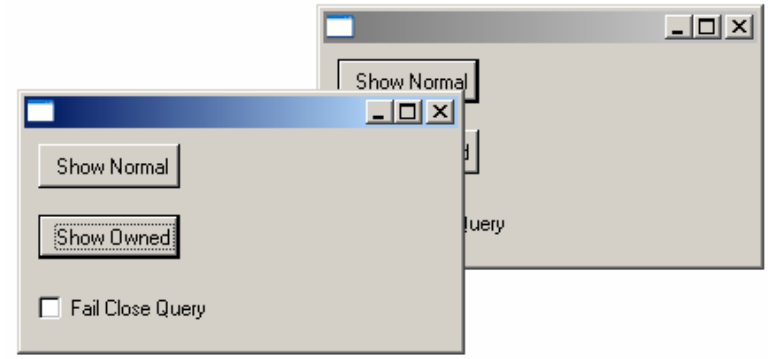


Use the Show Normal button:



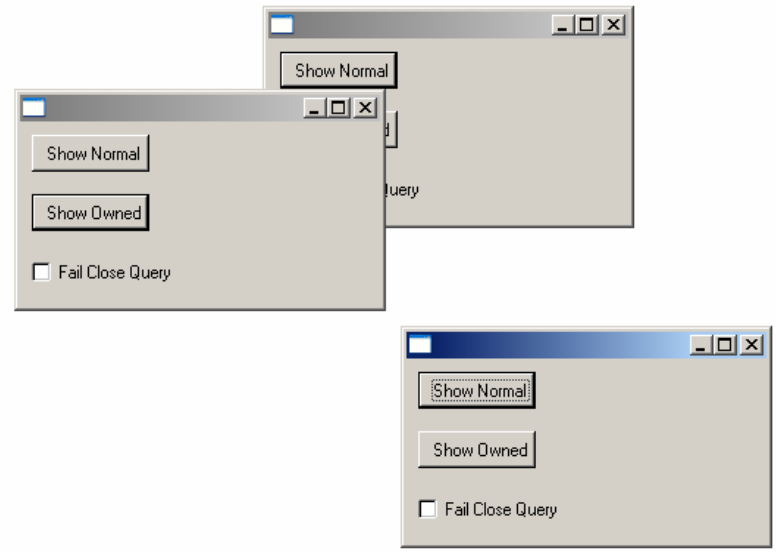
You will see a new task on your taskbar.

Now drag the latest form over the first form and activate the first form again by clicking its panel title:



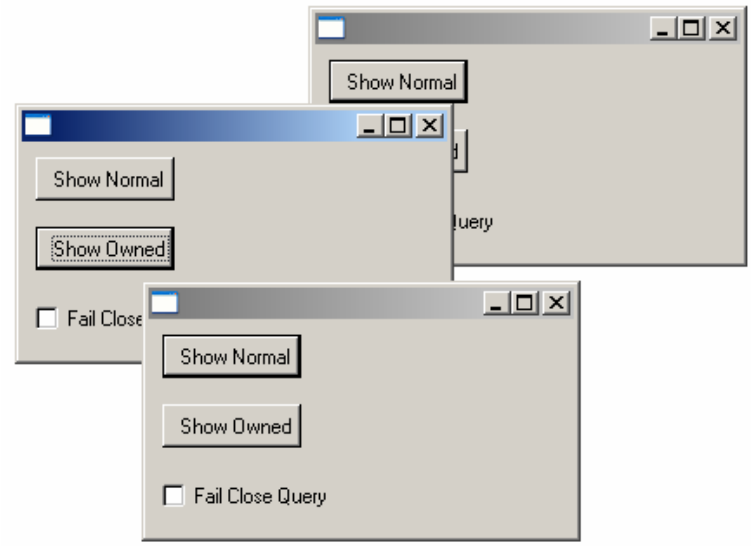
You will see that the second form (FormStyle Normal) will disappear behind the first form.

Use the Show Owned button:



Look at the task bar. There is no new task now, because this task is owned by its parent.

Again, drag this form over the first form and click on the panel title of the first form to activate it. The result:



The owned form will stay on top, although the parent form is the activated form.

Course Schedule January - June 2005

The course schedule for the first six months of 2005 is available now.

January 2005 until June 2005

Workshop	Nbr. of days	Jan	Feb	Mar	Apr	May	Jun
LANSA Basic	5 Days	3	21	-	18	-	13
LANSA Advanced	3-5 Days	-	7	-	-	31	-
LANSA for the Web	5 Days	-	-	-	25	-	27
Visual LANSA Basic	4 Days	24	-	8	-	10	-
Visual LANSA Advanced	3-5 Days	-	15	-	4	-	-
Visual LANSA Framework (Windows)	2 Days	10	28	-	-	2	20
Visual LANSA Framework (Web)	3 Days	12	-	2	-	4	22
Client-Server	1 Day	-	11	-	-	30	-
LANSA Integrator	2 Days	18	-	17	-	-	7
LANSA WAM	5 Days	31	-	21	-	23	-