http://www.lansa.com

LANSA

Newsletter
March 2007

Variant Handling

Function Libraries are a component that contains a set of routines defined using the *MthRoutine* command. These function library routines can be used in expressions.

There are function libraries for string, number, date and time manipulation (#PRIM_LIBS, #PRIM_LIBN, #PRIM_LIBD) but this is more simply accomplished using **Error! Reference source not found.**.
To handle objects of type *VARIANT, you need to use the variant function library #PRIM_LIBV.

Variants can contain any type of data (strings, integers, decimals, booleans, components). You can use variant functions for testing the kind of value contained in the variable and you can retrieve its value in converted forms like numbers, strings or booleans.

Variants make possible the generic processing of values regardless of their type. For example, the value of a grid cell cannot be known by the compiler before the application is executed. Therefore the Value parameter of grid EditorChanged and ItemChangedAccept events returns the value in the cell as a variant so that compiler errors about the declared type of the value do not occur. You then need to write the program in a way that understands what type(s) are in the grid. Also the EditorChanged and ItemChangedAccept events of tree and list views return the Value parameter as a variant.

Similarly, many ActiveX controls return and accept values as variants. You can also use variants in your own dynamic programs for generic processing of values regardless of their type:
Define_Com Class(*Variant) Name(#lclVariant)

In This Issue

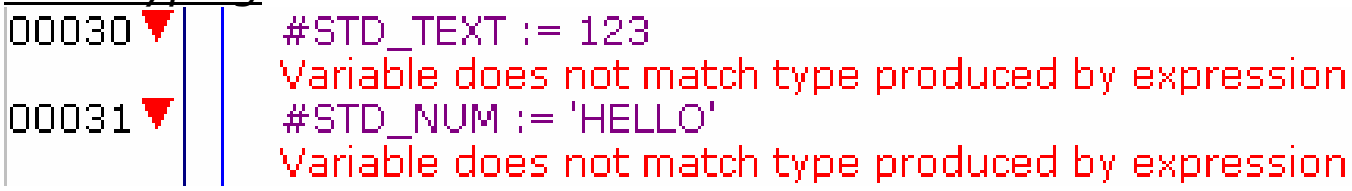
Using the *Variant class is the recommended approach. Alternatively, you can use primitive variant component #PRIM_VAR and its properties and methods. #PRIM_LIBV supports these functions:

VarType	VarIsString
VarIsBoolean	VarAsBoolean
VarIsEmpty	VarAsDecimal
VarIsNull	VarAsInteger
VarIsNullReference	VarAsReference
VarIsNumber	VarAsString
VarIsReference	

Variants

- A variant component variable can contain any type of data (strings, integers, decimals, booleans, components).
- A variant component has properties for testing the kind of value contained by the component.
- You can retrieve its value in converted forms like numbers, strings or booleans.
- Variants make it possible for components to give access to values whose type cannot be statically determined.

Field Typing



- All fields in LANSA are strongly typed.
- The type mismatch is immediately identified by the editor.
- Variants overcome this limitation.

Import

- You must import the #PRIM_LIBV for using with Variants:

```
IMPORT Libraries(#PRIM_LIBV)

DEFINE_COM Class(#PRIM_VAR) Name(#MYDATA)
#MYDATA := 'HELLO'
#STD_TEXT := VarAsString(#MYDATA)
#MYDATA := 123
#STD_NUM := VarAsDecimal(#MYDATA)
```

Function Libraries

- An alternative way of using Intrinsic Field Methods is to import function libraries.
- Function Libraries are a component that contain a set of routines defined using the MTHROUTINE command.

string	#PRIM_LIBS
number	#PRIM_LIBN
date and time	#PRIM_LIBD
*VARIANT	#PRIM_LIBV

Import Strings Library

If you import the #PRIM_LIBS for using with strings:
IMPORT Libraries(#PRIM_LIBS)

You can use the intrinsics in this form:

#STD_TEXT := Trim(#SURNAME)
#STD_TEXT := Lowercase(Trim(#SURNAME))

Support for SQL Server 2000 and SQL Server 2005 collections in Visual LANSA 11.3 (CU3)

The following support was delivered in 11.3 (CU3) and the text below was added to the EPC771 documentation. Refer to EPC771 (http://www.lansa.com/support/notes/epc/epc771.htm). The information provided below is very important to understand as it highlights that you cannot simply elect to upgrade an existing LANSA environment using an SQL Server database to 11.3 (CU3) and start using this new feature. A degree of planning and migration is required before an existing SQL Server database can avail of this newly supported SQL Server feature. Options and alternatives are discussed below.

==========

Support collections in SQL Server 2000 and SQL Server 2005

LANSA now allows the creation of collections with new SQL Server databases.

This is a critical change to SQL Server functionality. If you use SQL Server as your LANSA database and you created it using a version of LANSA prior to LANSA 11.3 CU3, and you have deployed applications that also use SQL Server, you MUST NOT use this new feature. You must continue to use a LANSA database created by prior versions of LANSA. This is because there is no migration possible from a database that does not support collections to one that does.

If you need to create a new installation of LANSA to maintain an application deployed prior to LANSA 11.3 CU3, then you must do one of the following:

- Install LANSA with a version of LANSA prior to 11.3 CU3;
- Use an existing SQL Server database that was created a version of LANSA prior to 11.3 CU3.

If you need to start using a new version of SQL Server to maintain an application deployed prior to LANSA CU3 - say upgrading from SQL Server 2000 to SQL Server 2005 - then you must do one of the following:

- Use SQL Server administration utilities to directly upgrade the SQL Server 2000 database to SQL Server 2005;
- Use SQL Server administration utilities to export the entire SQL Server 2000 set of schema and data and import it to the SQL Server 2005 database;
- Install LANSA to the SQL Server 2005 database with a version of LANSA prior to 11.0 CU3;

This feature is only available to be used for applications that have not yet been deployed with a version of LANSA prior to LANSA 11.3 CU3 (which may also include test environments that contain test data that must be retained).

If you are already using an Independent LANSA System created with a version of LANSA prior to 11.3 CU3 - one that does not have an iSeries Master - then you also CANNOT start using collections. This is because there is no migration possible.

If you are already using a Slave LANSA System created with a version of LANSA prior to 11.3 CU3, and that has not yet been eliminated because of one of the reasons described above, then provided you have checked all your changes into the iSeries you can use this new feature.

To use this new feature, create a new SQL Server database and then install LANSA to it. Collections will be automatically used.

Skype takes over the HTTP port and causes LANSA for WEB operations to fail

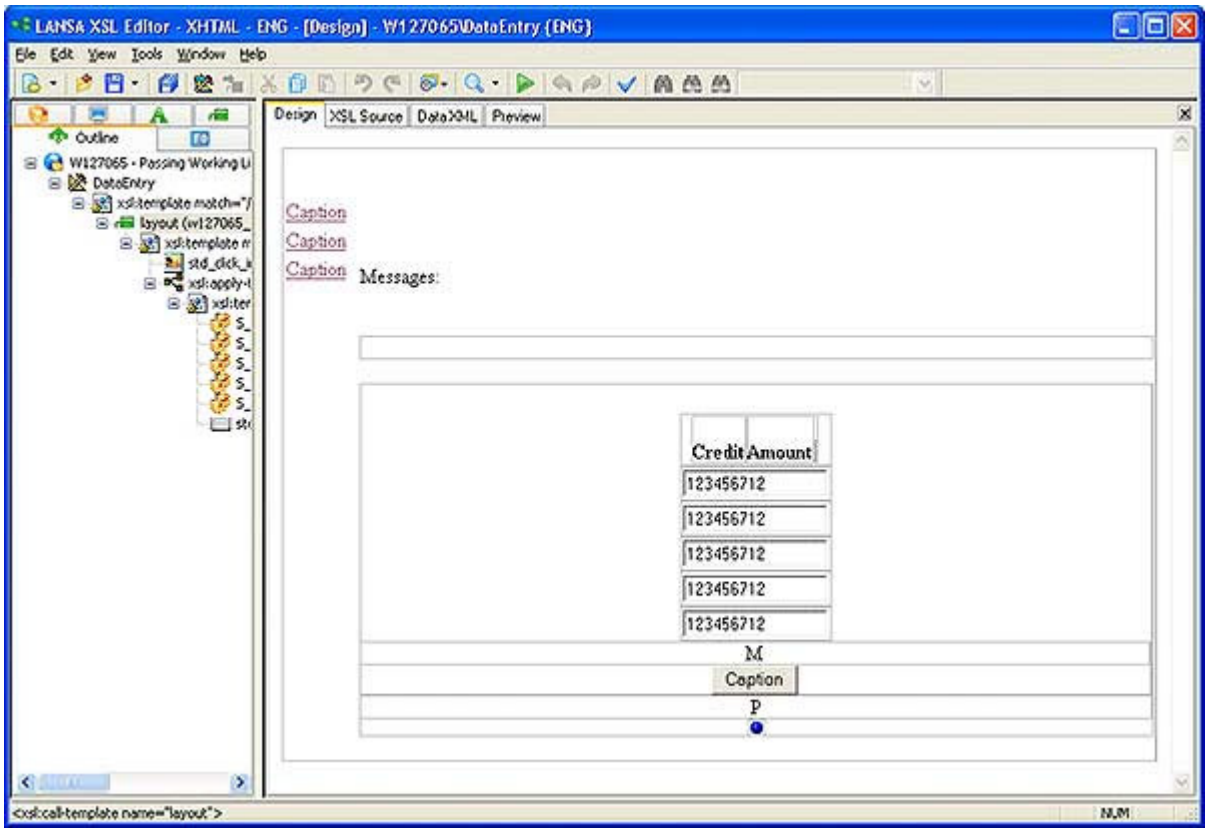
Installing Skype has been seen to interfere with existing LANSA for the web applications/operations. This is because Skype takes over the HTTP Port and confuses IIS.

What can happen is that if Skype is started before IIS, it will use port 80/443 before IIS can do so.
Which can mean that you won't get any page served by IIS (through these ports only, other ports used by IIS are not affected).
Even if IIS is set to start automatically as a service and Skype is to be started upon logon, the conflict can occur as it depends on timing.

To avoid this problem, you can disable the checkbox to use port 80 and 443 in Skype under Options -> Connection.

Layout and screen formatting disappears from the design screen in the XSL Editor

An issue has been found with one of the recent automatic updates to Internet Explorer, which causes the redraw of the XSL Editor design screen to fail. The effect of this is that at random times the design screen will lose all formatting and colors. This does not affect the execution of the WAM.



Workaround

It should be possible to recover from this by repeatedly pressing F5 to refresh the screen. In some cases the screen will not return to normal until you restart LANSA and the XSL Editor.

Since this Explorer update has already been superseded, it is not possible to remove this update on its own.

Solution

While the defect appears to be in the Update that Microsoft has supplied, a code workaround has been found that is scheduled to be addressed in an EPC for Visual LANSA 11.

Refer to the EPC (http://www.lansa.com/support/notes/epc/index.htm) information page for updates.

Scope *APPLICATION

The SCOPE parameter of the DEFINE_COM command has been enhanced to include option *APPLICATION.

All *APPLICATION variables are identified by variable name. Therefore, two different component classes can share a component instance simply by including a DEFINE_COM for the variable name and specifying a scope of *APPLICATION.

The first reference to an *APPLICATION scoped variable that is not *DYNAMIC will cause the component instance to be created. All other accesses retrieve that instance.

When a component instance at scope *APPLICATION is retrieved, the only checking performed is to ensure that the class of the component instance can be dynamically cast to the class specified on the variable's DEFINE_COM.

*APPLICATION variables are released when the application terminates. Care must be taken to ensure that the component classes used by an instance of a component at *APPLICATION scope are fully understood. All the component DLL's required to implement these component classes will remain in memory for the lifetime of the component instance and this could correspond to the lifetime of the application.

*The example below is created by Jurgen Rentinck from LANSA Amsterdam and shows how Scope *application works.*

A.
Create a new form called APP3. Copy/paste source below into it and compile the form.

```
* ************************************************
*
* COMPONENT:  STD_FORM
*
* ************************************************
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(104)
CLIENTWIDTH(509) HEIGHT(138) LEFT(481) TOP(202) WIDTH(517)

DEFINE_COM CLASS(#STD_TEXT.Visual) NAME(#STD_TEXT)
DISPLAYPOSITION(1) HEIGHT(19) LEFT(8) PARENT(#COM_OWNER)
TABPOSITION(1) TOP(32) USEPICKLIST(False) WIDTH(478)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Endroutine

End_Com
```

B.
Create a new form called APP2. Copy/paste source below into it and compile the form.

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(160)
CLIENTWIDTH(248) HEIGHT(194) LEFT(445) TOP(120) WIDTH(256)

DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2) CAPTION('show app3')
DISPLAYPOSITION(1) LEFT(78) PARENT(#COM_OWNER) TABPOSITION(1)
TOP(64)

Define_Com Class(#app3) Scope(*APPLICATION)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Endroutine

Evtroutine Handling(#PHBN_2.Click)
#app3.showform
Endroutine

End_Com
```

C.
Create a new form called APP1. Copy/paste source below into it and compile the form.

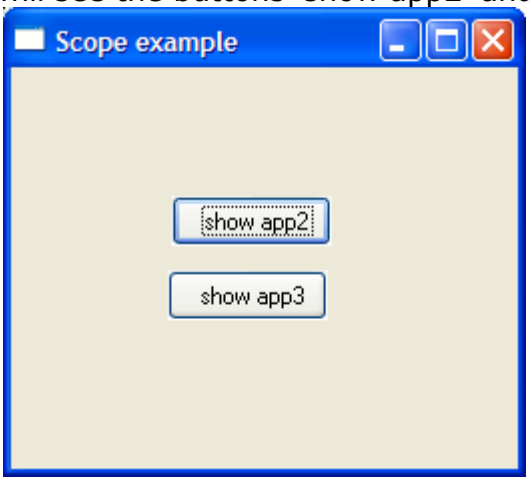
```
* ************************************************
*
* COMPONENT:  STD_FORM
*
* ************************************************
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(200)
CLIENTWIDTH(253) HEIGHT(234) LEFT(400) TOP(158) WIDTH(261)

DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1) CAPTION('show app2')
DISPLAYPOSITION(1) LEFT(80) PARENT(#COM_OWNER) TABPOSITION(1)
TOP(64)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2) CAPTION('show app3')
DISPLAYPOSITION(2) LEFT(78) PARENT(#COM_OWNER) TABPOSITION(2)
TOP(101)

DEFINE_COM CLASS(#app2) NAME(#APP2)

Define_Com Class(#app3) Scope(*APPLICATION)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Endroutine

Evtroutine Handling(#PHBN_1.Click)
#app2.showform
Endroutine

Evtroutine Handling(#PHBN_2.Click)
```

#app3.showform
Endroutine

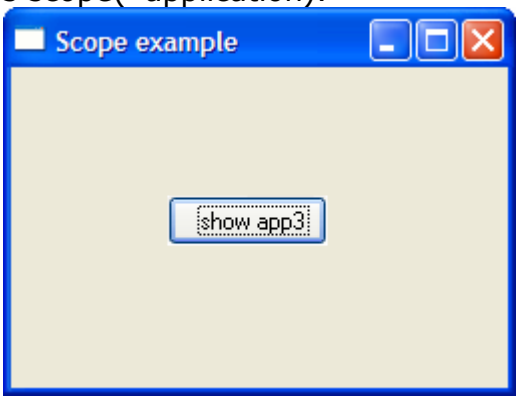
End_Com

To understand how Scope *Application works, follow the next steps:

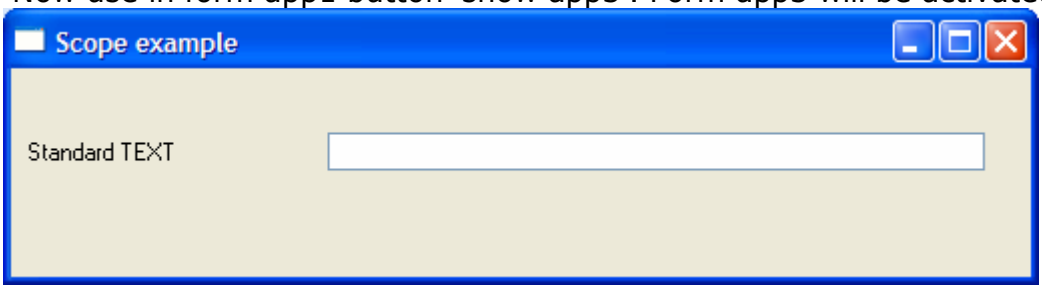
1. Start app1. You will see the buttons ‘show app2’ and ‘show app3’.



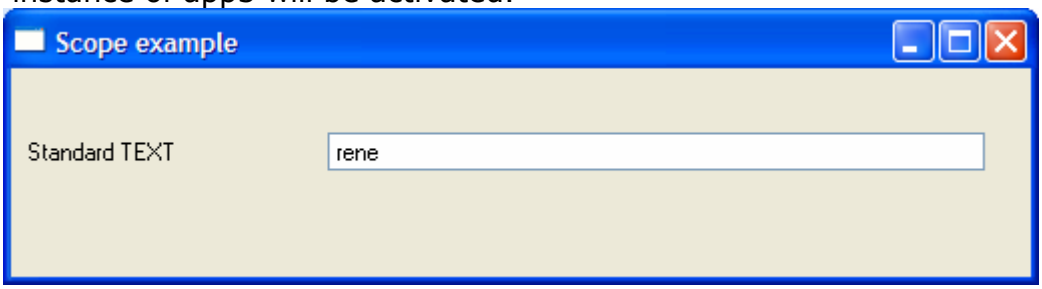
2. Use button ‘show app2’ to activate app2. In both form app1 as app2, form app3 is defined as scope(*application).



3. Now use in form app1 button ‘show app3’. Form app3 will be activated.



4. Now use in form app2 button ‘show app3’. No new instance of form app3 will be created, because of the scope *application setting.
5. When you give field std_text a value now in app3, close this form and restart app3 again in form app1 or app2, you will see that always the same instance of app3 will be activated.



How to use GZip in the Visual LANSA Framework ?

What is GZip?

GZIP file compression is a standard that most HTTP servers and Web browser support. Using it can significantly reduce the size of the HTML and JS files that are created as part of VLF web browser and RAMP applications.

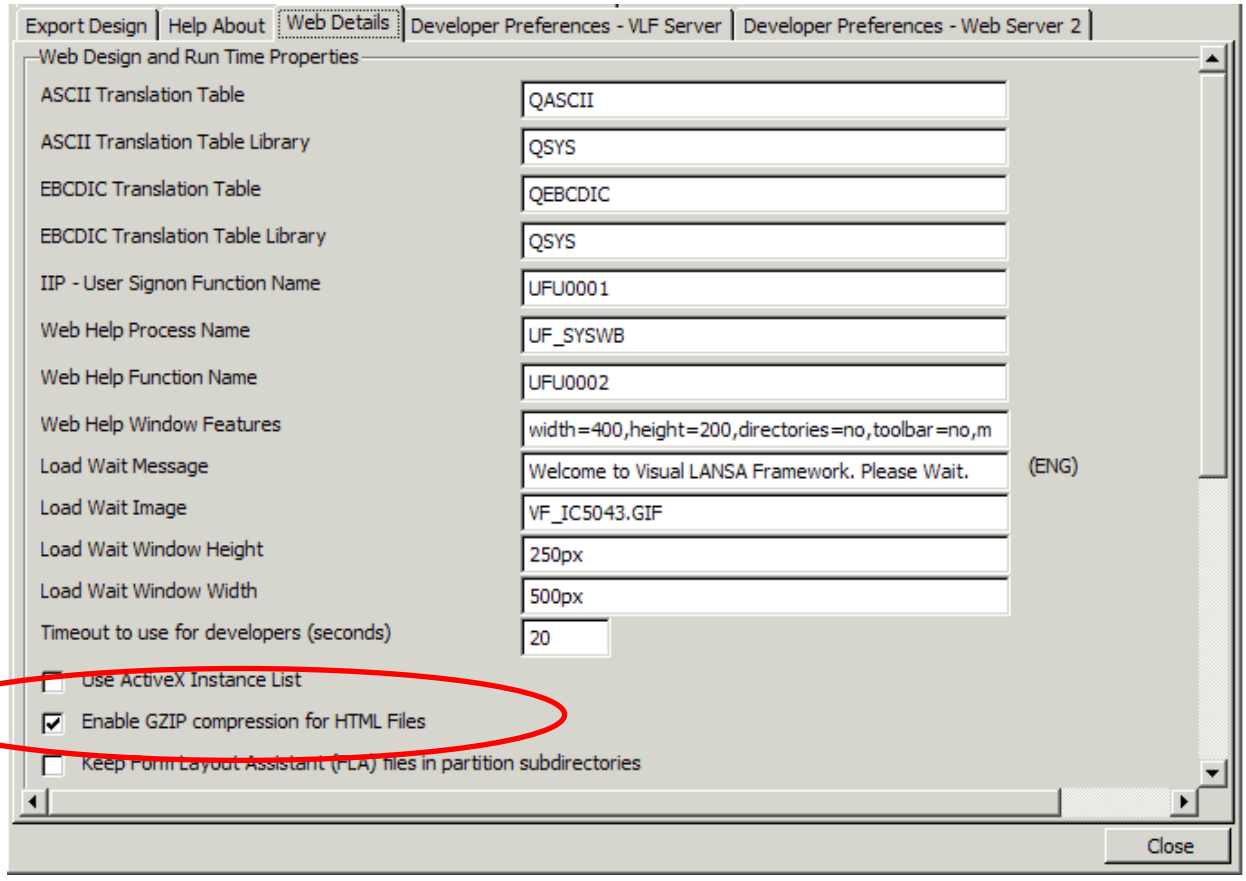
Framework applications are well suited to GZIP compression because the generated HTML and JS files are relatively static.

Using the option "GZIP Compression for HTML files" can create all newly generated HTML and JS files in both the normal uncompressed form and in the GZIP compressed form.

How to setup GZip in the Framework

In Visual LANSA Framework Designer mode , use the (Framework) -> (Properties) menu options and switch to the (Web Details) tab.

Enable the option called - "Use GZIP compression for HTML and JS Files" like below:



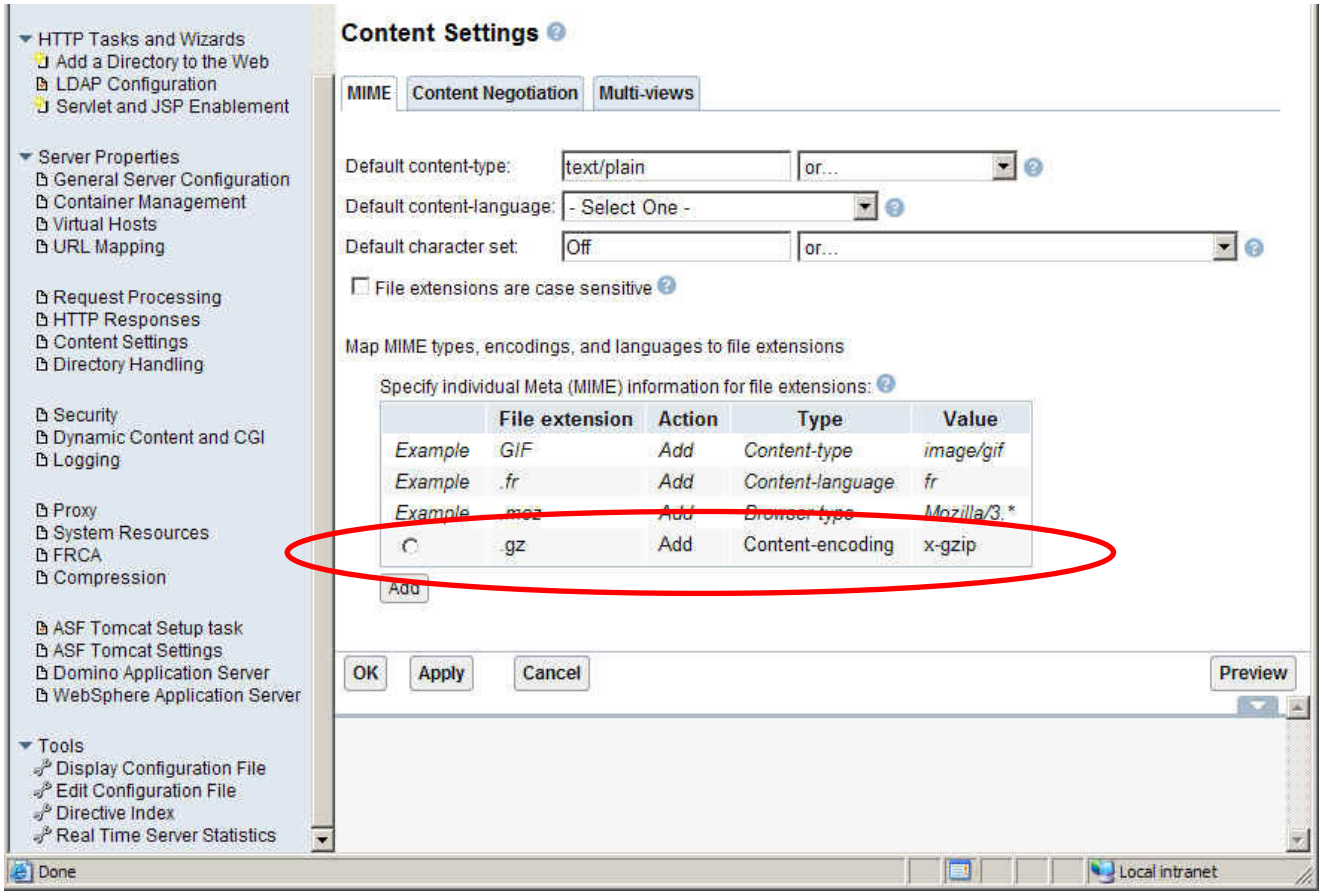
When this option is used all newly generated HTML and JS files will be created in BOTH the normal uncompressed format as well as in the GZIP compressed format (a compressed file has the same name as an uncompressed file ie. with the suffix of ".gz").

Some points to check:

1. This option is for an iSeries Web Server only (it should not be used with the Windows IIS Web Server).
2. When using this with an Apache Web Server ensure that it is configured correctly - so that when it returns a compressed GZIP file to the web browser it correctly informs the browser that the file is being delivered in compressed format.

How to setup in Apache Web Server

Start the Apache administrator, then invoke the "Content Settings" option and switch to the MIME tab:



Add file extension ***.gz*** (a GZIP file) so that it's ***Content-encoding*** is returned to the web browser as ***x-gzip***.

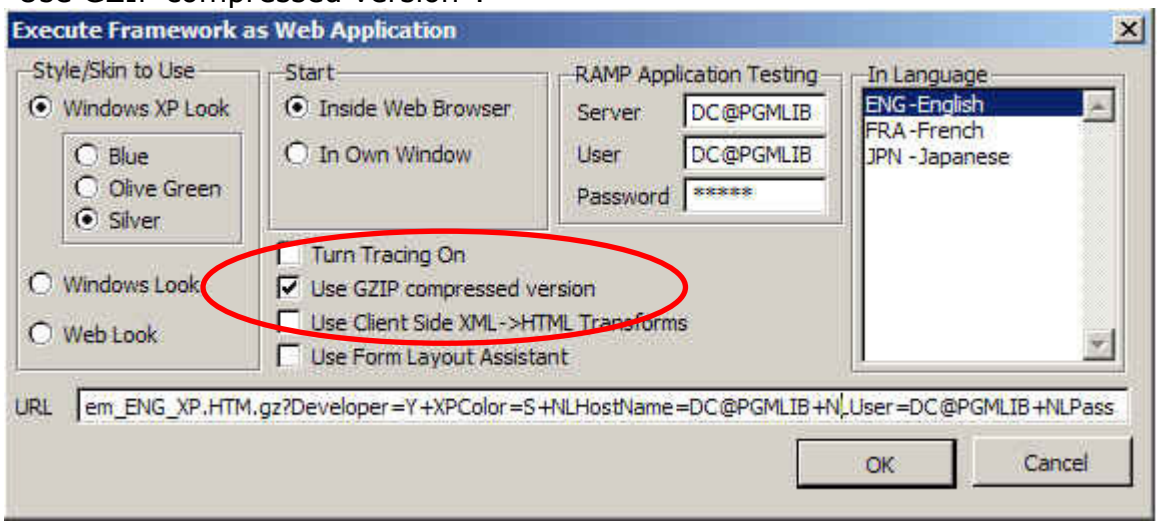
This will include --> AddEncoding x-gzip .gz into the APACHE configuration.

After adding this option to the MIME table you should shut down and restart the HTTP server instance for the change to take effect.

How to use GZip in the Framework

When you use the "Execute Framework as Web Application" dialog you may elect to execute either the normal uncompressed version or the compressed version of the Framework Application.

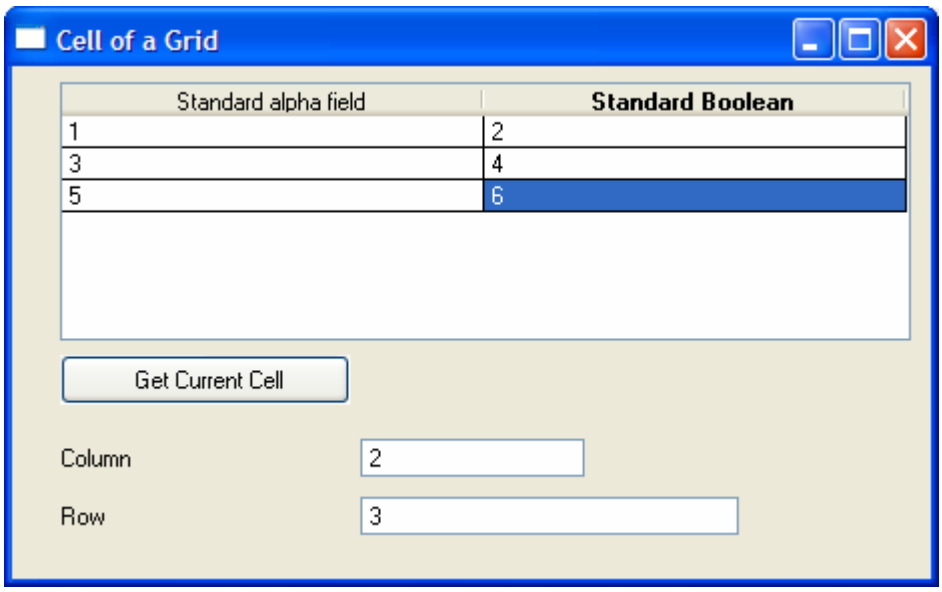
To use the GZIP version of the Framework Application - checked the option to "Use GZIP compressed version":



The cell of a Grid

(Thanks to Pascal Van Doorn from LANSA Amsterdam)

This little tool returns the current selected row and column of a Grid.



Copy/paste source below into a new form, compile and test it.

```
*
*
* COMPONENT:  STD_FORM
*
*
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(256)
CLIENTWIDTH(459) FORMPOSITION(ScreenCenter) HEIGHT(290) LEFT(502)
TOP(143) WIDTH(467)
DEFINE_COM CLASS(#PRIM_GRID) NAME(#GRID_1)
CAPTIONNOBLANKLINES(True) COLUMNSCROLL(False)
COMPONENTVERSION(1) DISPLAYPOSITION(1) HEIGHT(129) LEFT(24)
PARENT(#COM_OWNER) SHOWBUTTONSELECTION(True)
SHOWSELECTION(True) SHOWSELECTIONHILIGHT(False)
SHOWSORTARROW(True) TABPOSITION(1) TOP(8) WIDTH(425)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_1) DISPLAYPOSITION(1)
PARENT(#GRID_1) SOURCE(#STD_ALPHA) WIDTH(50)
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_2) DISPLAYPOSITION(2)
PARENT(#GRID_1) SOURCE(#STD_BOOL) WIDTH(50)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1) CAPTION('Get Current
Cell') DISPLAYPOSITION(3) LEFT(24) PARENT(#COM_OWNER)
TABPOSITION(2) TOP(144) WIDTH(145)
DEFINE_COM CLASS(#STD_NUM.Visual) NAME(#STD_NUM) CAPTION('Column')
DISPLAYPOSITION(2) HEIGHT(19) LABELTYPE(Caption) LEFT(24)
PARENT(#COM_OWNER) TABPOSITION(3) TOP(186) USEPICKLIST(False)
WIDTH(262)
DEFINE_COM CLASS(#STD_NUML.Visual) NAME(#STD_NUML) CAPTION('Row')
DISPLAYPOSITION(4) HEIGHT(19) LABELTYPE(Caption) LEFT(24)
```

```
PARENT(#COM_OWNER) TABPOSITION(4) TOP(215) USEPICKLIST(False)
WIDTH(339)

Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Change Field(#std_alpha) To('''1''')
Change Field(#std_bool) To('''2''')
Add_Entry To_List(#GRID_1)
Change Field(#std_alpha) To('''3''')
Change Field(#std_bool) To('''4''')
Add_Entry To_List(#GRID_1)
Change Field(#std_alpha) To('''5''')
Change Field(#std_bool) To('''6''')
Add_Entry To_List(#GRID_1)
Endroutine

Evtroutine Handling(#PHBN_1.Click)
Change Field(#std_num) To(#GRID_1.FocusCell.Column.Position)
Change Field(#STD_NUML) To(#GRID_1.FocusCell.Item.Entry)
Endroutine
End_Com
```

How do I improve performance of my Integrator Performance including SOAP Services?

The processing time for a SOAP Service request seems to be relatively long. If you are finding that a SOAP request takes more than 20 seconds or so then there might be performance tuning issues. Obviously there may be many factors causing the slow performance, like a large I/O to a file or a performance intensive logic and the like. Note that RDMLX processing will be relatively longer then RDML processing.

However, listed below are few important points to check for when attempting to improve performance of requests.

Note: These steps are not restricted to SOAP processing only. It may be also be used when having general performance issues with LANSA Integrator.

==========

Performance Tuning Considerations

1. Optimize (Optimise) the JAVA files to level 40. See online guide for more information on 'Optimize Java Service Manager (OPTJSM)'. The optimize operation has been seen to better the performance considerably.

2. If not required, turn Integrator tracing OFF.

3. Apply the latest CUME's and Java Group PTF's and Apache HTTP Server Group PTF's.

4. Run the Apache instance in CGIConvMode BINARY/BINARY.

5. Is there only one caller program calling the LANSA SOAP server ? You could pre-start some CGI jobs.

6. Is there high enough Activity Level for QBASE. The HTTP server and JSM by default will be running out of QBASE as well as other iSeries jobs. If the Activity Level is low only a small number of the jobs will be getting CPU time, you could be starving the jobs of CPU resources.

7. To improve the TCP/IP connection time between the JSM_OPEN and the JSM server make sure that a DNS name is used as the host name on the JSM_OPEN optional argument or the value in th e JSMCLTDTA dataarea. Also make sure that this DNS name is in the local HOST table.

   USE BUILTIN(JSM_OPEN) TO_GET(#JSMSTS #JSMMSG)
   Value
   Offset *...+....1....+....2
   0 'LOCALHOST:4560
   50 'JSM

8. To improve data transfer rates between the DCXS882X service program (BIF's) and the JSM server, the send and receive buffers can be configured.
   The latest DCXS882X BIF will also set the socket options for send and receive buffers to 128K before the connect. TCP/IP will then negotiate the connection between client and server.

manager.properties - no tcp.buffer properties (use the TCP/IP default values)

tcp.nodelay=*yes
tcp.buffer.send=131072
tcp.buffer.receive=131072

From trace file MANAGER.TXT
manager: tcp.port : 4560
manager: tcp.backlog : 20
manager: tcp.interface : *all
manager: tcp.nodelay : <null>
manager: tcp.buffer.send : <null>
manager: tcp.buffer.receive : <null>
manager: create manager server
manager: create socket address to listen on port 4560 across all interfaces
manager: bind to socket address
manager: start manager server
manager: server receive buffer size : 64000

Change TCP/IP Attributes (CHGTCPA)

Type choices, press Enter.

TCP keep alive . . . . . 120 1-40320, *SAME, *DFT
TCP urgent pointer . . . . . *BSD *SAME, *BSD, *RFC
TCP receive buffer size . . . . 64000 512-8388608, *SAME, *DFT
TCP send buffer size . . . . . 64000 512-8388608, *SAME, *DFT

manager.properties - tcp.buffer properties

tcp.nodelay=*yes
tcp.buffer.send=131072
tcp.buffer.receive=131072

From trace file MANAGER.TXT
manager: tcp.port : 4560
manager: tcp.backlog : 20
manager: tcp.interface : *all
manager: tcp.nodelay : *yes
manager: tcp.buffer.send : 131072
manager: tcp.buffer.receive : 131072
manager: create manager server
manager: create socket address to listen on port 4560 across all interfaces
manager: bind to socket address
manager: start manager server
manager: server receive buffer size : 131072
==========

** **Note**: Do not measure the response time on the first request. There is a big performance hit on the iSeries JAVA when loading classes for the first time.

iSeries VLF-WEB configuration and debugging tip

If you are having trouble configuring a VLF-WEB system an/or find that when you try to execute an application for the first time it fails for reasons that are not immediately obvious in the job log ..... then look for a file named VF_Server_Trace.dat in the root of the iSeries IFS.

This file should contain a lot more details about the system and the results of some specific configuration checks.

Even if the data content does not help you, it will certainly help the LANSA support person to identify the problem.

Tips

- If there is no VF_Server_Trace.dat file produced, it may be because the user profile you are executing the web application under is not authorized to write files into the root of the IFS.
- VF_Server_Trace.dat is formatted in EBCDIC, not ASCII, so it needs to be converted to view on a PC.

Using Internet Explorer 7 with Visual LANSA Framework Web Applications

LANSA previously provided instructions on how to allow your Visual LANSA Framework Web Applications to execute with IE7. These instructions relate to the BETA 2 version of IE7 available at the time.

Now that IE7 is generally available, support for IE7 is provided via a hotfix for Visual LANSA Framework at EPC793 level only. To use the IE7 support hotfix, you must have applied EPC793.

Contact your local LANSA vendor to request this hotfix.

The next version of the Visual LANSA Framework will fully support IE7.

Summary

Visual LANSA Framework	*IE7 (Beta)*	*IE7 (GA)*
EPC785	Yes*	No
EPC793	Yes	No**

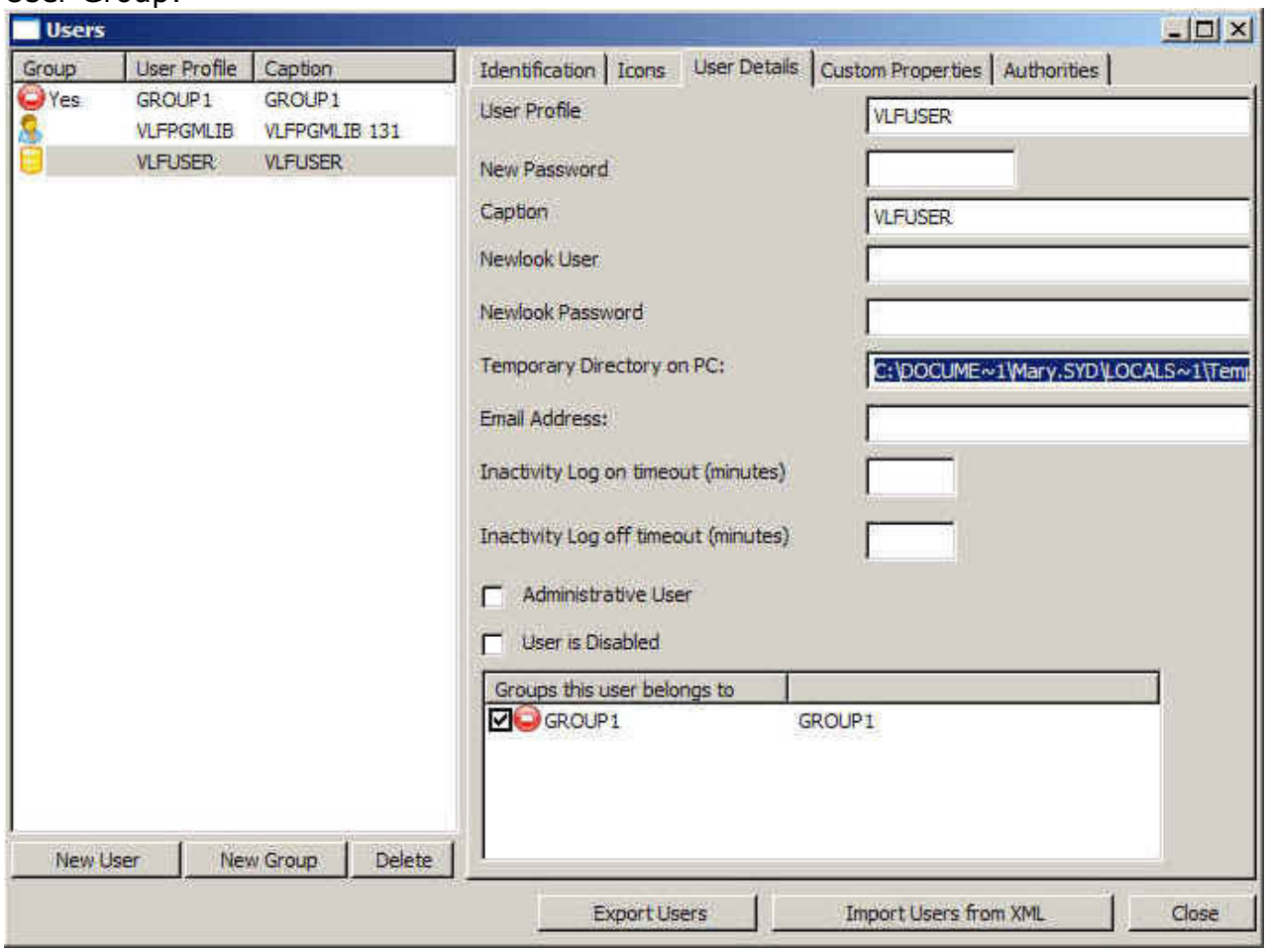
* Available via Instructions. Refer to Using VLF with IE7 Beta 2 (http://www.lansa.com/support/tips/t0398.htm)

** IE7 support is provided via a hotfix for Visual LANSA Framework at EPC793 level only.

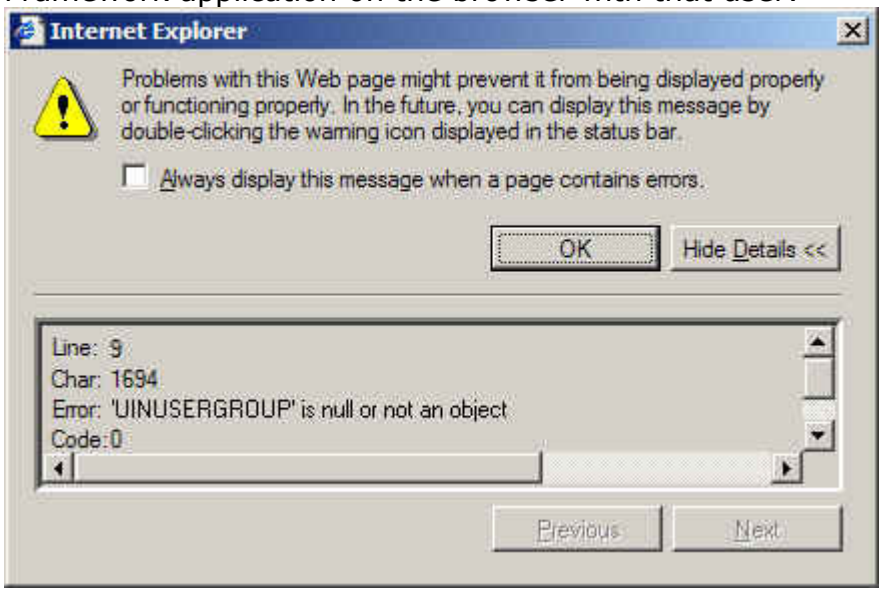
UINUSERGROUP error VLF

An error *'UINUSERGROUP' is null or not an object'* is displayed when running a Visual LANSA Framework application on the browser.

If you defined a Framework User with authorities that belongs to a Framework User Group:



You might experienced this error at the login window when running the Framework application on the browser with that user:



This can happen when logging in with a user that belongs to a User Group defined in the Framework. The error will not occur when logging in with a user that does not belong to a User Group defined in the Framework. If you are experiencing this issue please contact LANSA Support for a hot-fix. The official solution will be available in future versions of the Framework.

MCH3601 QC2POSIX error on V5R1 after applying EPCs to LANSA 11.3 (CU3)

The following applies to V5R1 only.

Any EPC post EPC771 (EPC771 is also referred to as CU3) should not be applied to a LANSA 11.3 system if this LANSA system is on OS/400 V5R1. This specifically means you cannot apply EPC790 or higher to LANSA 11.3. By applying post CU3 EPCs to a LANSA 11.3 environment, you may generate a connection error when connecting to V5R1 via the LANSA Listener.
The error in the iSeries joblog is:

*MCH3601 Escape 40 29/01/07 15:26:07 QC2POSIX QSYS *STMT QC2POSIX QSYS *STMT*
From module . . . . . : QC2PLOCL
From procedure . . . . . : _C_load_DB_ctype
Statement . . . . . : 2355
To module . . . . . : QC2PLOCL
To procedure . . . . . : _C_load_DB_ctype
Statement . . . . . : 2355
Message . . . . : Pointer not set for location referenced.
Cause . . . . . : A pointer was used, either directly or as a basing pointer, that has not been set to an address.

- This error is specific to V5R1.
- V5R1 is no longer supported by IBM.
- This error does not occur on current IBM supported OS/400 versions.
- This error can occur when using any or all of the following LANSA features
  - LANSA Host Monitor
  - LANSA Superserver
  - Model B LANSA for the web
  - LANSA Open
  - LANSA Client
  - any other LANSA product or feature that use the LANSA listener

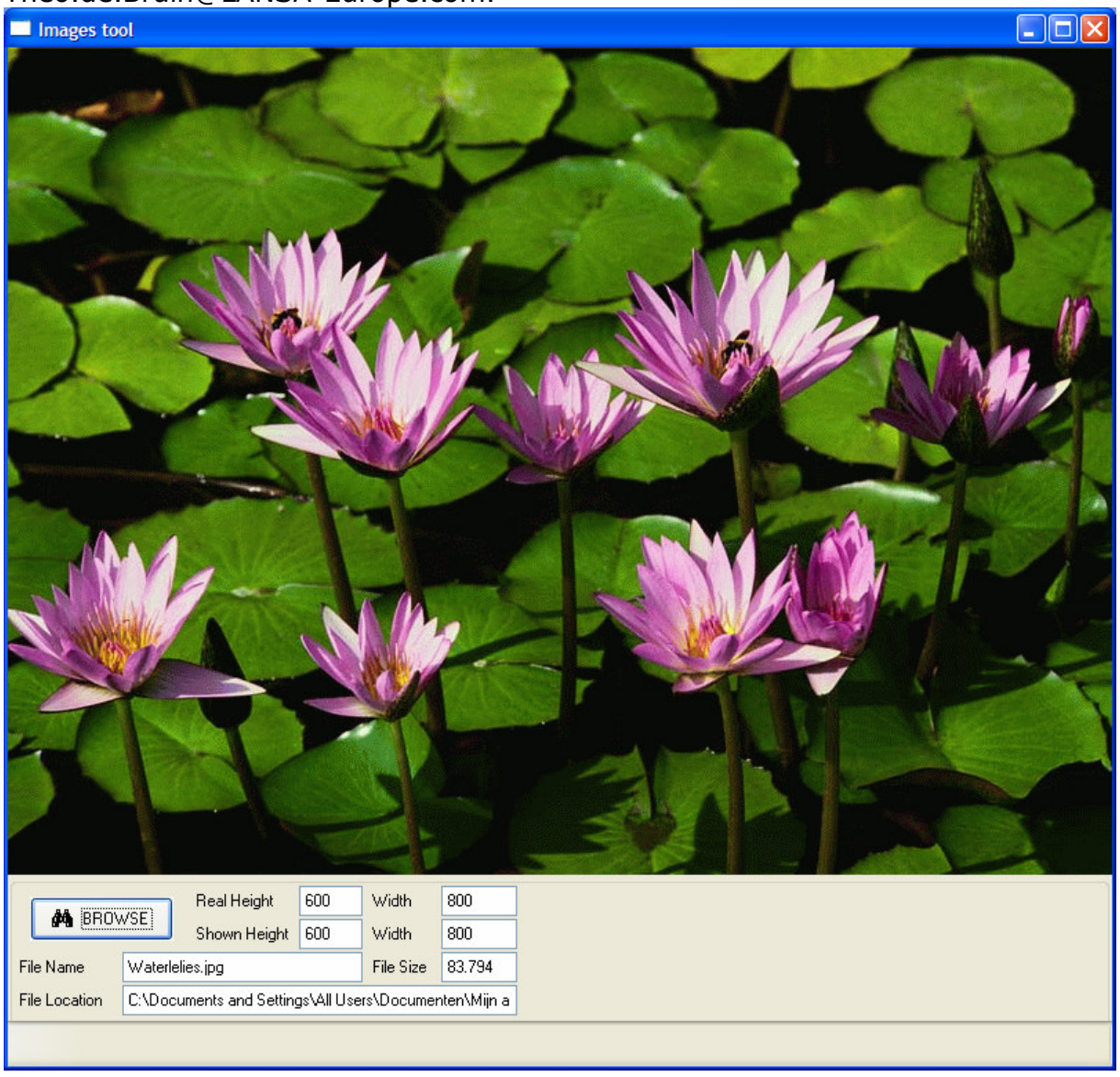
Resolution

There is no circumvention available from LANSA for this error. If you generate this error after applying a post CU3 EPC, LANSA recommends that you restore from your last backup prior to applying the post CU3 EPCs. **You must upgrade to a supported version of OS/400 before applying any post CU3 EPC.**

Images Viewer

(Thanks to Theo de Bruin from LANSA Amsterdam)

This tool is using the Windows Dialog to select an image, which is shown in its original size, or resized to the maximum resolution, without loosing its aspect ratio. It works on every screen resolution.
Please feel free to alter this to your own needs. Suggestions can be mailed to Theo.de.Bruin@LANSA-Europe.com.



To make a compile possible of the source below, you need to create two new field components into the Repository.

- #STD_NUM10, Signed, length 10, EditCode J, attribute RB
- #IMAGE, Blob, attribute *ASQN and *LC, default *SQLNULL

Copy/past source below into a new form:

```
************************************************
*
* COMPONENT :  IMG_VIEWER
* Created on:  26-09-2006
* Created by:  Theo de Bruin - LANSA ltd., Amsterdam
* General   :  Form is using the Windows Dialog to
*              select an image, which is shown in its
*              original size, or resized to the max.
*              resolution, without loosing its aspect
*              ratio.
* Amendments:  Please feel free to alter this to your
*              own needs. Suggestions can be mailed
*              Theo.de.Bruin@LANSA-Europe.com
************************************************
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('TBN Image Viewer')
CLIENTHEIGHT(433) CLIENTWIDTH(374) FRAMESTYLE(Single) HEIGHT(467)
LAYOUTMANAGER(#ATLM_1) LEFT(484) TOP(8) WIDTH(382)
DEFINE_COM CLASS(#PRIM_IMGE) NAME(#IMGE_1) DISPLAYPOSITION(1) HEIGHT(294)
LEFT(0) PARENT(#COM_OWNER) TABPOSITION(1) TOP(0) WIDTH(374)
DEFINE_COM CLASS(#STD_QSEL.Visual) NAME(#FIL_PATH) CAPTION('File Location')
DISPLAYPOSITION(5) HEIGHT(22) LABELTYPE(Caption) LEFT(8) MARGINLEFT(75)
PARENT(#GPBX_1) TABPOSITION(6) TOP(80) USEPICKLIST(False) WIDTH(361)
DEFINE_COM CLASS(#STD_NUM.Visual) NAME(#FIL_SIZE) CAPTION('File Size')
DISPLAYPOSITION(3) HEIGHT(22) LABELTYPE(Caption) LEFT(264) MARGINLEFT(50)
PARENT(#GPBX_1) TABPOSITION(4) TOP(56) USEPICKLIST(False) WIDTH(105)
DEFINE_COM CLASS(#STD_TEXTS.Visual) NAME(#FIL_NAME) CAPTION('File Name')
DISPLAYPOSITION(1) HEIGHT(22) LABELTYPE(Caption) LEFT(8) MARGINLEFT(75)
PARENT(#GPBX_1) TABPOSITION(2) TOP(56) USEPICKLIST(False) WIDTH(249)

* File Open Dialog

Define_Com Class(#PRIM_APPL.ICommonDialogFileOpen) Name(#openFileDlg)
Reference(*DYNAMIC)
DEFINE_COM CLASS(#PRIM_GPBX) NAME(#GPBX_1) DISPLAYPOSITION(2) HEIGHT(139)
LEFT(0) PARENT(#COM_OWNER) TABPOSITION(2) TABSTOP(False) TOP(294)
WIDTH(374)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2) CAPTION('BROWSE')
DISPLAYPOSITION(6) HEIGHT(32) IMAGE(#VB_SEARCH) LEFT(16) PARENT(#GPBX_1)
TABPOSITION(8) TOP(16) WIDTH(104)
DEFINE_COM CLASS(#PRIM_STBR) NAME(#STBR_1) DISPLAYPOSITION(7) HEIGHT(33)
LEFT(0) MESSAGEPOSITION(1) PARENT(#GPBX_1) TABPOSITION(5) TABSTOP(False)
TOP(106) WIDTH(374)
*
DEFINE_COM CLASS(#STD_NUM10.Visual) NAME(#IMG_HGHT) CAPTION('Real Height')
DISPLAYPOSITION(4) HEIGHT(22) LABELTYPE(Caption) LEFT(136) MARGINLEFT(75)
PARENT(#GPBX_1) TABPOSITION(3) TOP(8) WIDTH(121)
DEFINE_COM CLASS(#STD_NUM10.Visual) NAME(#IMG_WDTH) CAPTION('Width')
DISPLAYPOSITION(9) HEIGHT(22) LABELTYPE(Caption) LEFT(264) MARGINLEFT(50)
PARENT(#GPBX_1) TABPOSITION(9) TOP(8) WIDTH(105)
DEFINE_COM CLASS(#STD_NUM10.Visual) NAME(#PNL_HGHT) CAPTION('Shown
Height') DISPLAYPOSITION(2) HEIGHT(22) LABELTYPE(Caption) LEFT(136)
MARGINLEFT(75) PARENT(#GPBX_1) TABPOSITION(1) TOP(32) WIDTH(121)
DEFINE_COM CLASS(#STD_NUM10.Visual) NAME(#PNL_WDTH) CAPTION('Width')
DISPLAYPOSITION(8) HEIGHT(22) LABELTYPE(Caption) LEFT(264) MARGINLEFT(50)
PARENT(#GPBX_1) TABPOSITION(7) TOP(32) WIDTH(105)
```

```
* Layout MAnagement
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1) ATTACHMENT(Center)
MANAGE(#IMGE_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_2) ATTACHMENT(Bottom)
MANAGE(#GPBX_1) PARENT(#ATLM_1)

* Work Fields
DEFINE #SFACTORH DECIMALS(6) REFFLD(#STD_NUM) DEFAULT(1)
DEFINE #SFACTORW DECIMALS(6) REFFLD(#STD_NUM) DEFAULT(1)

* Size difference between  Form and Image
DEFINE_COM CLASS(#STD_NUM) NAME(#B_Width)
DEFINE_COM CLASS(#STD_NUM) NAME(#B_Height)

*Fileinfo
DEFINE FIELD(#OV_PATH) TYPE(*CHAR) LENGTH(256)
DEFINE FIELD(#OV_RETC) TYPE(*CHAR) LENGTH(2)
DEFINE FIELD(#OV_ERRN) TYPE(*DEC) LENGTH(15) DECIMALS(0)
DEFINE FIELD(#OV_NAME) TYPE(*CHAR) LENGTH(100)
DEFINE FIELD(#OV_PREFIX) TYPE(*CHAR) LENGTH(12)
DEFINE FIELD(#OV_SUFFIX) TYPE(*CHAR) LENGTH(3)
DEFINE FIELD(#OV_DATE) TYPE(*CHAR) LENGTH(8)
DEFINE FIELD(#OV_TIME) TYPE(*CHAR) LENGTH(6)
DEFINE FIELD(#OV_ISDIR) TYPE(*CHAR) LENGTH(1)
DEFINE FIELD(#OV_SIZE) TYPE(*DEC) LENGTH(9) DECIMALS(0) EDIT_CODE(3)
DEF_LIST NAME(#FLIST) FIELDS(#OV_NAME #OV_PREFIX #OV_SUFFIX #OV_DATE
#OV_TIME #OV_SIZE #OV_ISDIR) TYPE(*WORKING) ENTRYS(5000)


*=====================================================
=============
Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
*determmine the difference between Form and Image
#B_Height := #com_owner.height - #IMGE_1.height
#B_Width:= #com_owner.width - #IMGE_1.width

Endroutine

Mthroutine Name(ChooseProductImage) Help('Pick a product image via Windows xplorer')
Define_Map For(*result) Class(#STD_QSEL) Name(#o_FileName)
Define Field(#boolRes) Type(*BOOLEAN)

Invoke Method(#sys_appln.CreateFileOpenDialog) Result(#openFileDlg)
#openFileDlg.Title := 'Select Image to be displayed'
#openFileDlg.AddFilter( 'JPG Images (*.jpg)' '*.jp*' )
#openFileDlg.AddFilter( 'GIF Images (*.gif)' '*.gif' )
#openFileDlg.AddFilter( 'BMP Images (*.bmp)' '*.bmp' )
#openFileDlg.AddFilter( 'All files  (*.*)' '*.*' )
#openFileDlg.FilterIndex := 1
#openFileDlg.HideReadOnly := false
#openFileDlg.ExplorerStyle := false
#openFileDlg.MultiSelect := false

Invoke Method(#openFileDlg.Show) Okpressed(#boolRes) Formowner(#com_self)
If (#boolRes)
```

```
* OK BUTTON PRESSED
#o_FileName := #openFileDlg.File
Else
* CANCEL BUTTON PRESSED
#o_FileName := *blanks
Endif
Endroutine

Evtroutine Handling(#PHBN_2.Click #IMGE_1.Doubleclick)
* set the product image filename to the file name of an image chosen by the user.
#Image := #com_owner.ChooseProductImage

* If the user chose an image, assign it to the #img_1 component (and check for EXIF)
If (#IMAGE *NE *BLANKS)
#IMGE_1.FileName #STD_QSEL := #Image.FileName
If Cond(#IMGE_1.Format = UNKNOWN)
Use Builtin(OV_Message_Box) With_Args("Unknown image format, or JPG contains EXIF
data")
Else
#Com_Owner.SetSize Height(#IMGE_1.ImageHeight) Width(#IMGE_1.ImageWidth)
* GET the file size
#FIL_PATH := #STD_QSEL.leftmost(#IMGE_1.Filename.lastpositionof('\'))
#FIL_NAME := #STD_QSEL.Substring((#IMGE_1.Filename.lastpositionof('\') + 1) 100)
USE BUILTIN(OV_FILE_SERVICE) WITH_ARGS(GET_DIR #FIL_PATH) TO_GET(#OV_RETC
#OV_ERRN #FLIST)
LOC_ENTRY #FLIST WHERE('#OV_NAME = #FIL_NAME')
#FIL_SIZE := #OV_SIZE
Endif
Endif
Endroutine

MTHROUTINE Name(Setsize)
Define_Map For(*Input) Class(#Std_Num) Name(#Height)
Define_Map For(*Input) Class(#Std_Num) Name(#Width)

#SFACTORH #SFACTORW := 1
#IMG_HGHT #IMGE_1.Height := #Height
#IMG_WDTH #IMGE_1.Width := #Width

#com_owner.height := #IMG_HGHT + #B_Height
#com_owner.width := #IMG_WDTH + #B_Width
* determine if picture is larger than resized screen
IF ('#IMGE_1.imagewidth > (#com_owner.width - #B_Width)')
#SFACTORW := ((#com_owner.width - #B_Width) / #imge_1.imagewidth)
ENDIF
IF ('#IMGE_1.imageheight > (#com_owner.height - #B_Height)')
#SFACTORH := ((#com_owner.height - #B_Height) / #imge_1.imageheight)
ENDIF
* Keep ratio
IF ('#SFACTORW < #SFACTORH')
#SFACTORH := #SFACTORW
ELSE
#SFACTORW := #SFACTORH
ENDIF
* if larger,shrink form
IF ('#SFACTORH < 1')
set #com_owner height((#IMG_HGHT * #SFACTORH) + #B_Height) top(1)
ENDIF
```

```
IF ('#SFACTORW < 1')
set #com_owner width((#IMG_WDTH * #SFACTORW) + #B_Width) left(1)
ENDIF
#PNL_Wdth := #IMGE_1.width
#PNL_Hght := #IMGE_1.height
ENDROUTINE

End_Com
```