http://www.lansa.com

LANSA

Newsletter
February 2006

Award for LANSA

Best Industry Optimized Solution

Did you know that your LANSA is an award-winning company? ☺

Perhaps, most interestingly, this award is for one of our newer products, LANSA Data Secure Direct, and in area that we think has real growth potential in Europe.
This is the 1st time LANSA wins this prestigious IBM Beacon award in his history.

This award announcement is on:
http://www-1.ibm.com/partnerworld/pwhome.nsf/weblook/pub_awards_winner.html

To help enterprises meet new global product-data-synchronization requirements, LANSA developed its LANSA Data Sync Direct solution to allow manufacturers, distributors and retailers to link product data between their systems and a certified data pool within the Global Data Synchronization Network (GDSN). The solution uses the information-aggregation power of DB2 and the open-standards-based foundation of WebSphere to enable transactions to be transmitted directly from a company server to a data pool without the need for third-party exchanges. When Glazer's Wholesale Drug Company was looking for a GDSN solution to link its system with its business partners' systems, it turned to LANSA Data Sync Direct to interlock manufacturing, procurement, sales and distribution, and thereby reduced costs, quickened transactions and improved customer service.
The judges appreciated that LANSA Data Sync Direct devised a solution with an end-to-end portfolio of IBM WebSphere and DB2 products that also included eServer iSeries, pSeries and xSeries components.

In This Issue

LANSA2005
GreenPieces

GreenPieces are documents that you can download that supplement the LANSA documentation to help you have the very best experience of LANSA 2005. Check back here from time-to-time for updates and additions.

Upgrading to LANSA 2005
If you are an existing LANSA user, this information will help you understand the benefits of this major release, and to plan for a smooth, successful upgrade that realizes those benefits.

- Top 5 Reasons for a 5250 Developer to switch to using the Visual LANSA Integrated Development Environment (PDF 1.67MB)
- Top 7 Reasons to upgrade to LANSA 2005 from Visual LANSA Version 10.0 (PDF 1.44MB)
- Stepping up to LANSA 2005 Frequently Asked Questions (PDF 162KB)

Understanding LANSA 2005 for Developers
LANSA 2005 unveils an almost entirely new Integrated Development Environment (IDE) in which

- the interface has been dramatically revised and modernized
- many power and productivity features have been added or substantially improved
- integration throughout the developer environment has been achieved to an extent not seen in previous versions.

These articles are about the new IDE. The first give some insights into the workings of the IDE and how to get the best out of it as a developer. The second describes how we went about using LANSA itself to deliver this new IDE.

- Getting the most from the LANSA 2005 Integrated Development Environment (PDF 175KB)
- The LANSA 2005 IDE: A case study in modernizing a C/C++ application using Visual LANSA (PDF 689KB)

The LANSA 2005 product set uses the term full RDMLX to refer to an optional extended set of capabilities and syntax for the LANSA repository and RDML language. These articles give a thorough introduction to the new capabilities and syntax associated with RDMLX in LANSA 2005:

- Who Put the 'X' in RDMLX? (PDF 220KB)
- Understanding Full RDMLX language syntax (PDF 183KB)
- Understanding RDMLX on iSeries (PDF 354KB)

LANSA 2005 introduces a comprehensive set of field types. Not only does this introduce much-requested types such as date and time, but it also enhances LANSA's ability to extend and enhance existing applications, through the use of OTHER files. These articles provide more information about using the new field types:

- LANSA Field Type Quick Reference (PDF 138KB)
- Getting started with BLOBs and CLOBs (PDF 180KB)
- Getting started with DateTime (PDF 197KB)

'Read Only' shared drives on Windows Servers stops Visual LANSA Network Clients connecting to server

In the V11.0 *Installing LANSA on Windows* guide, the section *1.1 BEFORE You Install or Upgrade LANSA* makes the following point:
If you are performing a Visual LANSA Client Only Network Install, ensure that you have mapped to a drive on the server where Visual LANSA is installed. Also check that you have access rights to the mapped drive.

This step has two parts:

1. Sharing the LANSA directory on the Windows server.
2. Creating a map for this shared directory

In older Windows operating systems, sharing a drive or directory would be created with "Full Control" by default. In newer Windows operating systems (e.g. Windows 2000 Server, Windows 2003 server), when you share a drive or directory, the share is created with only "Read" permission by default. This causes the Visual LANSA network client to not be able to connect to the Visual LANSA server. When attempting to run Visual LANSA Development Environment from a network client, nothing happens.

The solution is to amend the windows server shared directory to have "Full Control" which then allows the network client to connect.

Code Fragment Virtual Field

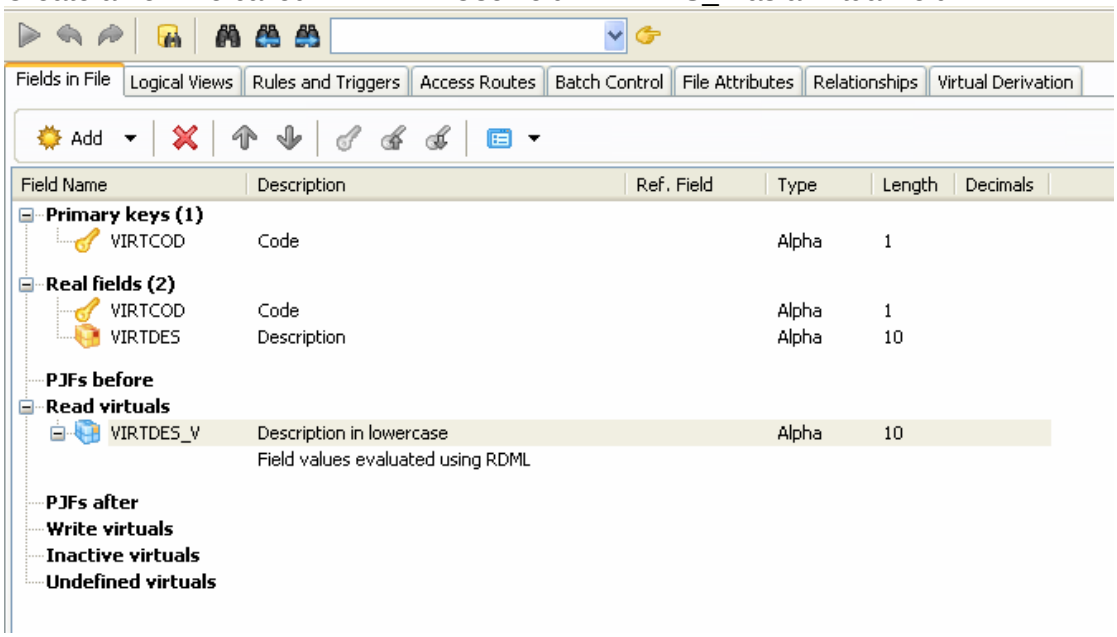
The code fragment virtual field allows the user to specify RDMLX code to populate a virtual field when reading from a file, and to populate a real field when writing to a file.
A code fragment does not support the full range of RDML/X commands. In Version 11, the user is limited to constructs such as If, Case, Dountil, Dowhile, Change, Assign.
At execution time, the code fragment will have read only access to all fields in the file. You may use trigger functions for complex coding of virtual fields or to update fields in a file.

Below an example of this new Virtual Field type.

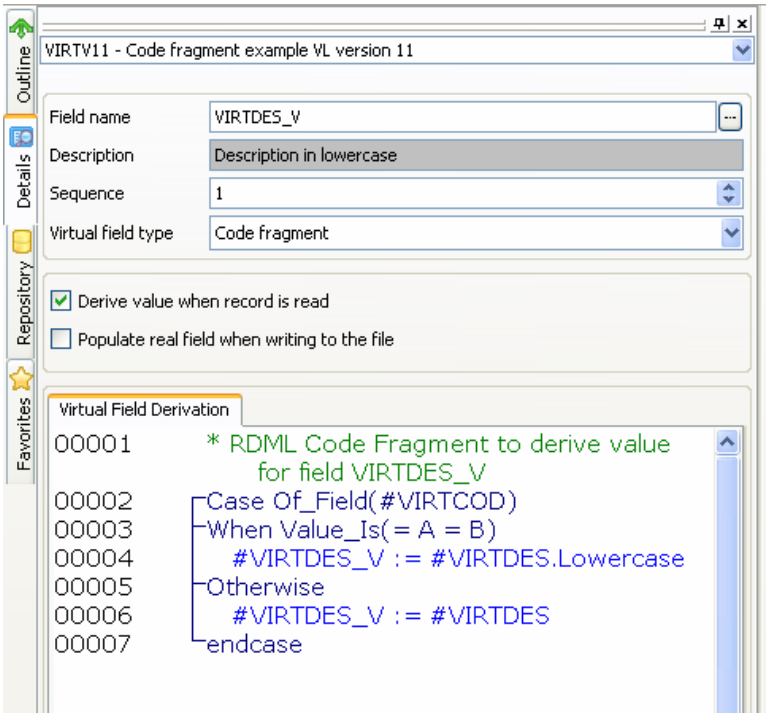
Create three new fields in the LANSA Repository:

VIRTCOD	Code	Alpha	1
VIRTDES	Description	Alpha	10
VIRTDES_V	Description in lowercase	Alpha	10

Create a new file called VIRTV11. Use field VIRTDES_V as a virtual field.



Use field VIRTDES_V as a Code fragment Virtual field type. When code field VIRTCOD has a value A or B, the virtual field will be filled with the description field value in lowercase, in all other cases the virtual field will be filled with the description field value.



Now create a new form component and copy/paste source below into it:

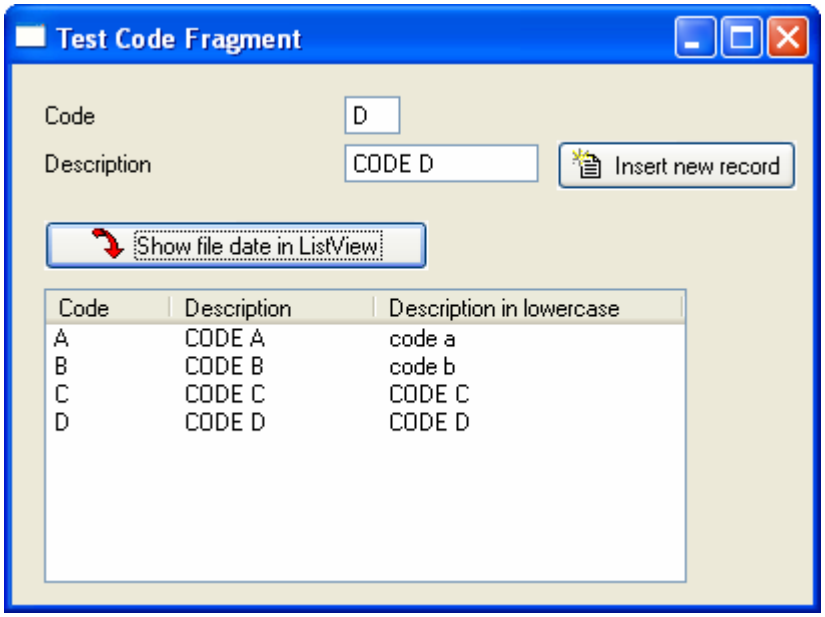
```
**************************************************
*
* COMPONENT:  STD_FORM
*
**************************************************
FUNCTION OPTIONS(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(270) Clientwidth(400) Height(304)
Left(539) Top(191) Width(408)
Define_Com Class(#VIRTDES.Visual) Name(#VIRTDES) Displayposition(1) Height(19)
Left(16) Parent(#COM_OWNER) Tabposition(1) Top(40) Usepicklist(False) Width(247)
Define_Com Class(#VIRTCOD.Visual) Name(#VIRTCOD) Displayposition(2) Height(19)
Left(16) Parent(#COM_OWNER) Tabposition(2) Top(16) Usepicklist(False) Width(178)
Define_Com Class(#PRIM_PHBN) Name(#PHBN_1) Caption('Insert new record')
Displayposition(3) Image(#VB_NEW) Left(272) Parent(#COM_OWNER) Tabposition(3)
Top(38) Width(120)
Define_Com Class(#PRIM_LTVW) Name(#LTVW_1) Componentversion(2)
Displayposition(4) Fullrowselect(True) Height(147) Left(16) Parent(#COM_OWNER)
Showsortarrow(True) Tabposition(4) Top(112) Width(321)
Define_Com Class(#PRIM_LVCL) Name(#LVCL_1) Displayposition(2) Parent(#LTVW_1)
Source(#VIRTDES) Width(32)
Define_Com Class(#PRIM_LVCL) Name(#LVCL_2) Displayposition(1) Parent(#LTVW_1)
Source(#VIRTCOD)
Define_Com Class(#PRIM_LVCL) Name(#LVCL_3) Displayposition(3) Parent(#LTVW_1)
Source(#VIRTDES_V) Width(42) Widthtype(Remainder)
```

```
Define_Com Class(#PRIM_PHBN) Name(#PHBN_2) Caption('Show file date in ListView')
Displayposition(5) Image(#VB_ARROWC) Left(16) Parent(#COM_OWNER) Tabposition(5)
Top(78) Width(192)

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

EVTROUTINE HANDLING(#PHBN_1.Click)
insert *all virtv11
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_2.Click)
clr_list #ltvw_1
select #ltvw_1 virtv11
add_entry #ltvw_1
endselect
ENDROUTINE
END_COM
```

Start the form to test how it works:



Recursive Triggers in an iSeries RDMLX Partition can loop indefinitely instead of aborting with a recursive error message

Before using triggers in your files, refer to *Triggers - Some Do's and Don'ts* section of the *iSeries User Guide* in the LANSA Online documentation.

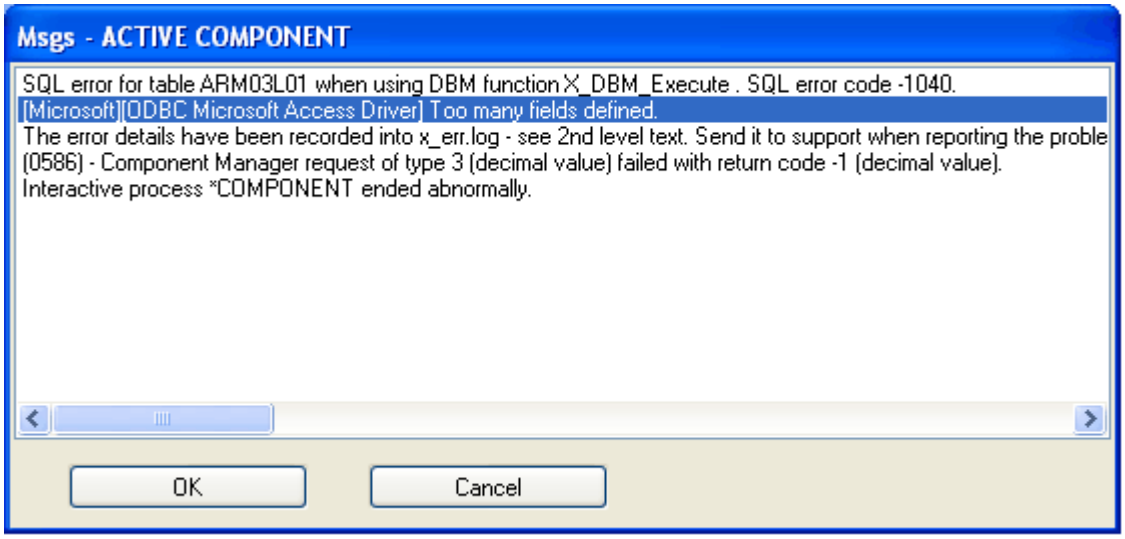
When using triggers, do not have any rules or validations along its access routes that may potentially lead back to any of the triggering paths.

For example, if FileA has a trigger/access route associated with FileB and File B has a trigger/access route linking back to FileA, then you have a case of a recursive operation. In a non-RDMLX partition, a recursive error will be generated but in an RDMLX partition, this recursive operation will cause the process to loop and take up resources..

The recursive behaviour will be addressed such that this will not cause the process to hang and the process will abort with an error message.

'Too Many Fields Defined' error on an UPDATE query when using a Microsoft Access Other file with more than 127 fields

An error will occur when attempting to update a Microsoft Access file that has been loaded into Visual LANSA as an 'OTHER' file and contains more than 127 fields. The error is as follows: "Too many fields are defined" as shown below:



There is a limitation in Microsoft Access on the number of fields that can be contained within a table. During an update query, a field for the original value is created and one for the updated value exists as well, ultimately causing two fields to exist during an update. It follows that if there are more than 127 fields MS Access's limit of 255 fields is then reached causing the error mentioned above.

Performing Install or Upgrade of LANSA for iSeries causes objects to be not restored depending on system value QALWOBJRST

Performing an install or upgrade of LANSA for iSeries (irrespective of version) may result in objects not being restored.

Job log would contain this:
"770 objects restored. 1084 not restored to DC@PGMLIB"

CPD373F Diagnostic 30 05/08/27 15:45:07.041656 QSRRSDEQ QSYS 09BB QSRRSDEQ QSYS 09BB
Message . . . . : BI@P002 in library DC@PGMLIB with adopt authority attribute not restored.
*Cause . . . . . : BI@P002 type *PGM in library DC@PGMLIB has the adopt authority attribute.*

*The Allow Object Restore (QALWOBJRST) system value has been set to not allow the restore of objects that have the adopt authority attribute. Recovery . . . : Set the Allow Object Restore (QALWOBJRST) system value to one of the following and try the request again: -- *ALL to allow the restore of all objects regardless of any security sensitive attributes. -- *ALWPGMADP to allow the restore of programs that adopt authority.*

These errors are generated if system value QALWOBJRST has been set not to allow restore of objects that adopt authority.

Prevention

Check the system value QALWOBJRST and change this system value to *ALL before the install/upgrade to avoid the install or upgrade ending in error.

Solution

In a case where you have attempted an install or upgrade and generated the failure messages in the joblog, you will need to change this system value QALWOBJRST to *ALL and re-run the Upgrade again.
Before you can re-run you will have to restore all the LANSA libraries and delete the new IFS folder specific to the LANSA system that was created. If you took the default name for the IFS then it will be something like this '/LANSA_dc@pgmlib'.

info4iSeries-expo - Netherlands

Date: 12 and 13 April 2006
Place: IJsselhallen in Zwolle

As you can see in the title of this article, the info4iSeries-expo is an iSeries Solutions Event. Everything during this expo has to do with the IBM iSeries (AS/400, eServer i5) platform. When your company already has an iSeries-server, or will buy a new one, and you want to make use of all new platform features, then you must be part of this expo!

LANSA will be there at stand number 25.
When you want to visit this expo, please send an email to:
Jan de Putter jan.de.putter@lansa-europe.com

For more information, visit the website at: http://www.info400.com/info4iseries/

Passing information to a WAM application by HTTP variables

The FUNCPARMs option of LANSA for the Web is not supported in WAMs, but there is a similar feature as well.

Arguments are added to the URL like this:

http://server/cgi-bin/lansaweb?webapp=a+webrtn=r+part=dem+field(XXXX)=ZZZZ

You would need a web map for input for field XXXX, and the field would retrieve the value ZZZZ.

Development Tools Race

Date: 23 and 24 February 2006

The Development Tools Race is organized in the Netherlands by Software Release Magazine since 2001. Development Tools 2006 already has 14 teams now. More important: every team represents all important technologies for the development of applications. Experienced Java-teams and .Net-teams, but also the 4GL's are part of this race.

The only iSeries Development Tool during this Development Tools Race is LANSA. This is the first time that we are part of this game.

LANSA will be represented by Pascal van Doorn and Jurgen Rentinck. In the next newsletter we will tell you all about the results.

For more information, visit: http://www.developmenttools.nl

Considerations for RDMLX on iSeries

1. Files

1.1 Before you do anything:

- It is ***not*** necessary to RDMLX-enable your files in order to use them with RDMLX-enabled functions and components. All RDMLX-enabled functions and components can access RDML files (but see 1.2 below)
- There is ***no*** advantage to RDMLX-enabling files (irrespective of what functions will call them) unless you wish to make use of RDMLX file definition features such as:
    - New field types
    - Virtual fields defined by RDMLX code fragments

1.2 If you want to use RDML files with RDMLX functions and components in an RDMLX-enabled partition:

- You must check-out and check-in the files in order to build the 'C' language OAM. (Note: this is ***not*** the same as RDMLX-enabling the file.)
- It is probably most convenient to do this when you RDMLX-enable a partition.
- You should check for incompatible features in your file definitions before proceeding. For example, virtual fields derived using RPG code will not work through the 'C' OAM.
- This can only be done through VL because the 'C' OAM code is generated by the VL client on check-in. The code cannot be generated on the iSeries.
- When you have done this, an RDML file will have (on iSeries) an IOM (generated in RPG) *and* an OAM (generated in 'C'). These are independent – the OAM does not call the IOM but is a complete implementation of the file access code. RDMLX functions and components will access the file through the OAM. RDML functions will access the file through the IOM.
- If you have not generated the 'C' OAM for an RDML file, RDMLX functions and components that access the file will receive a run-time error indicating a service program not found condition.

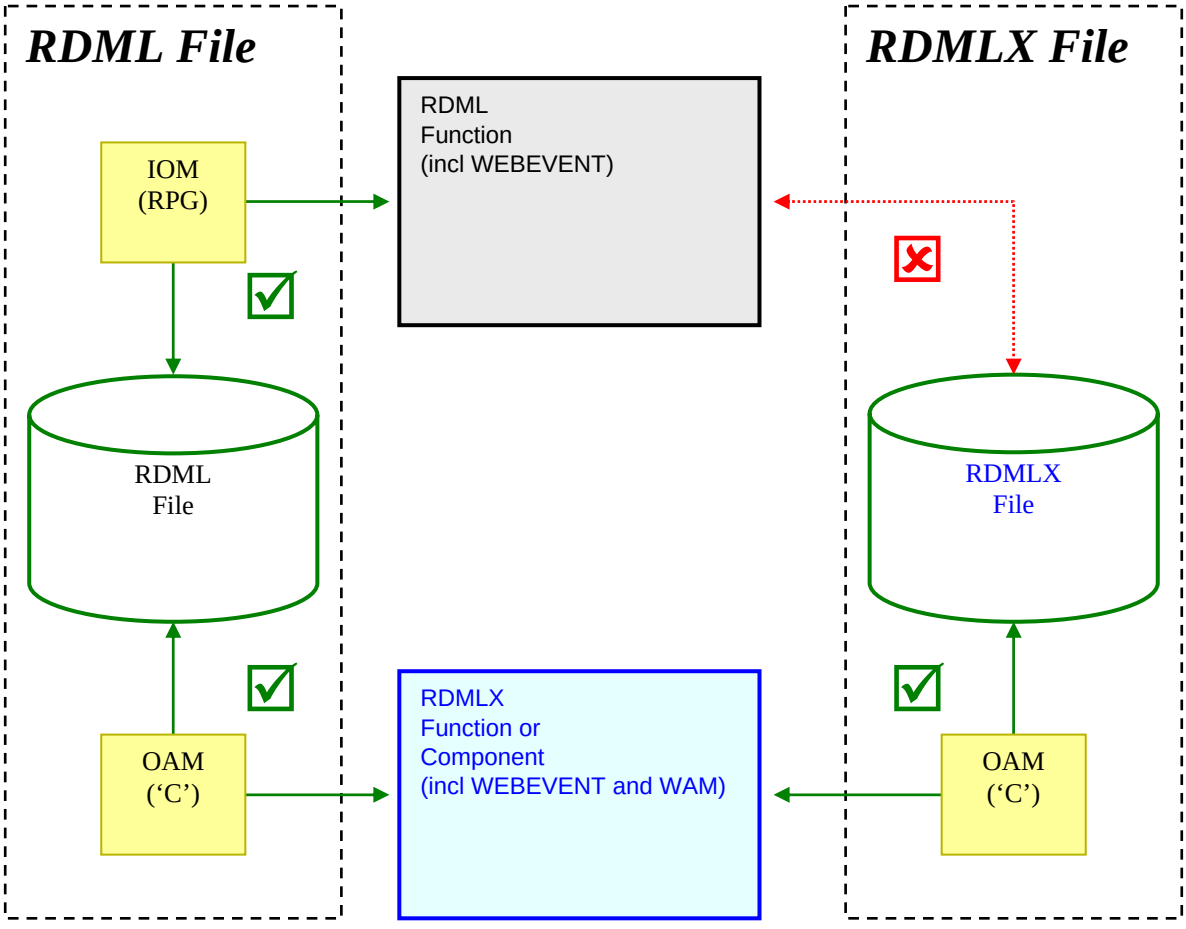
1.3 If you want to RDMLX-enable your files:

- You ***only*** need to do this if you wish to make use of RDMLX file definition features such as:
  - New field types
  - Virtual fields defined by RDMLX code fragments
- If you do this RDML functions will ***no longer*** be able to access the file directly.
- For existing files, you should only ever do this if you are absolutely sure you will never need to access the file in RDML (for example, in 5250 functions).
- RDMLX files do not have an RPG I/O Module. They only have the 'C' language OAM.

1.4 Accessing files in RDML and RDMLX Functions and Components

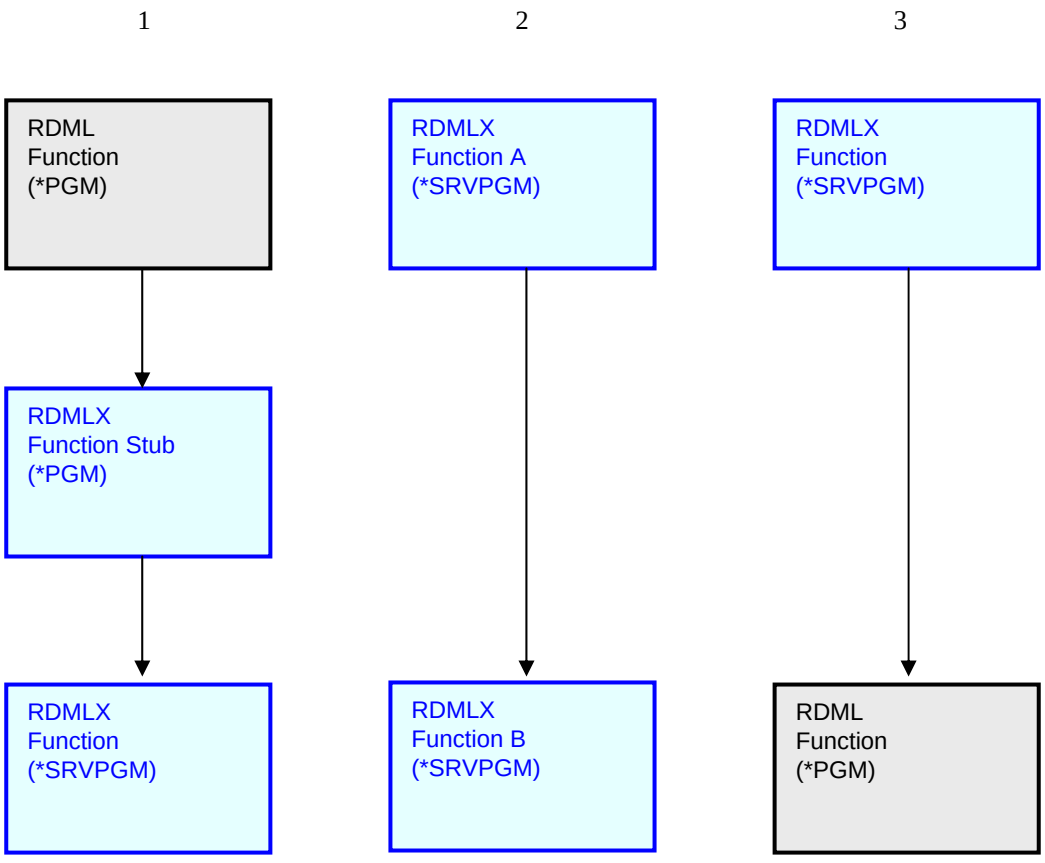
As shown below, LANSA functions and components can access any file defined in the repository, *with one notable exception*:

- ☒ An RDML function or component cannot directly access an RDMLX file.



2. Functions (and components)

- You can call freely between RDML and RDMLX functions.
- In order to minimise the impact on existing applications it is ***not*** necessary to recompile RDML functions in order to make this work.
- When you compile an *RDMLX* function, two objects are created:
  - A service program (*SRVPGM) that contains the implementation of the function code
  - A "stub" 'C' program (*PGM) that calls the function in the *SRVPGM and is generated purely in order to permit calls from RDML functions generated in RPG
- When an *RDMLX* function calls an *RDML* function, it looks first for an RDMLX function (implemented in a *SRVPGM). (This may cause a message in the joblog to the effect that the service program was not found). If the RDMLX version is not found, then it will call the RDML version (implemented in a *PGM object)
- Conversely, when an *RDML* function calls an *RDMLX* function it *only* looks for a *PGM object. When the target is an RDMLX function, the *PGM object is a "stub" that simply calls the RDMLX function implementation in the *SRVPGM.
- The diagram below shows the program objects involved in:
  1. an RDML function calling an RDMLX function
  2. an RDMLX function calling another RDMLX function
  3. an RDMLX function calling an RDML function



Technique to convert a string to "words" (i.e. separates text based on a space)

This tip shows how to separate text based on the spaces with in the text. The technique displays some simple string handling techniques available in Visual LANSA version 11. This is best explained with an example.

- Create a form named "mytip" or something similar (do not name it "words")
- Cut and paste the following code into the form and compile and run
- Enter a sentence into the field at the top of the form
- Each word is separated out and added to the list
- Refer to in-line code comments to gain an understanding of how this is used

Code:

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Convert a sentence to individual words')
CLIENTHEIGHT(313) CLIENTWIDTH(695) LEFT(74) TOP(126) WIDTH(703)
DEFINE_COM CLASS(#STD_QORD.Visual) NAME(#Sentence) CAPTION('Enter sentence')
DISPLAYPOSITION(1) HEIGHT(25) LABELTYPE(Caption) LEFT(8) MARGINLEFT(100)
PARENT(#COM_OWNER) TABPOSITION(1) TOP(8) USEPICKLIST(False) WIDTH(681)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#LV_Words) COLUMNBUTTONHEIGHT(20)
COMPONENTVERSION(2) DISPLAYPOSITION(2) FULLROWSELECT(True) HEIGHT(255) LEFT(8)
PARENT(#COM_OWNER) SHOWSORTARROW(True) TABPOSITION(2) TOP(40) WIDTH(680)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_1) CAPTION('Words') CAPTIONTYPE(Caption)
DISPLAYPOSITION(1) PARENT(#LV_Words) SOURCE(#STD_QSEL) WIDTH(21)
WIDTHTYPE(Remainder)

Evtroutine Handling(#Sentence.Changed) Options(*NOCLEARMESSAGES *NOCLEARERRORS)

Clr_List Named(#lv_words)

For Each(#Word) In(#Com_owner.StringToWords( #Sentence ))

#Std_Qsel := #Word

Add_Entry To_List(#lv_words)

Endfor

Endroutine

Mthroutine Name(StringToWords)
Define_Map For(*Input) Class(#Prim_alph) Name(#String)
Define_Map For(*Result) Class(#prim_lcol<#Prim_alph>) Name(#Words) Pass(*by_reference)

Define_Com Class(#prim_nmbr) Name(#FirstSpace)

Set_Ref Com(#words) To(*create_as #prim_lcol<#prim_alph>)
```

```
* Loop until nothing left in the string to process
Dountil (#String.cursize = 0)

* Drop any leading and trailing spaces
#String := #String.trim

* Look for the first space in the string
#FirstSpace := #string.PositionOf( ' ' )

* If no space found, must be a whole word
If (#FirstSpace = 0)

#FirstSpace := #String.cursize

Endif

#Words.insert( #Com_owner.AddWord( #String.Substring( 1 #FirstSpace ) ) )

* Remove the word from the string
#String := #String.DeleteSubstring( 1 #FirstSpace )

Enduntil

Endroutine

Mthroutine Name(AddWord) Access(*private)
Define_Map For(*Input) Class(#Prim_alph) Name(#Word)
Define_Map For(*Result) Class(#Prim_alph) Name(#Result) Pass(*by_reference)

Set_Ref Com(#Result) To(*create_as #Prim_alph)

#Result.value := #Word

Endroutine

End_Com
```