Newsletter
June 2015

LongRange on entire Windows 8.1 platform

Upcoming version of LANSA's LongRange will enable IBM i developers to build native apps across the entire Windows 8.1 platform – phones, tablets and desktop/laptop computers

LongRange is strengthening its mobile offering by adding support across the entire Windows 8.1 platform. Not only does this include Windows phones and tablets, but also desktop/laptop computers. Available this summer, LongRange's ability to create brand new Windows Modern UI (formerly Metro-style) desktop applications is a first for the IBM i community.

Build native apps across the entire Windows 8.1 platform!

In This Issue

"Using our unique set of solutions and development tools, LANSA is able to deploy new business applications faster than our competition," said Steve Gapp, president LANSA Americas. "With the upcoming release of LongRange, customers can extend the mobile apps they developed for Apple and Android to the Windows platform – without having to write any additional code. Even more impressive, they can run their LongRange apps on Windows 8.1 PCs."

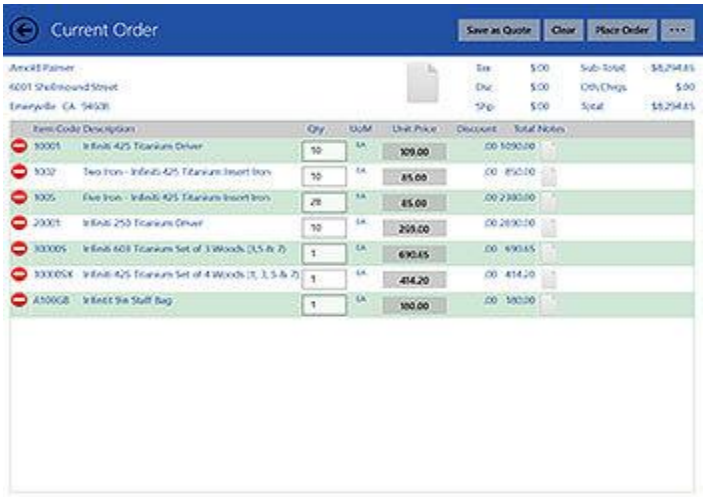
These capabilities are attractive to BWI Companies Inc., a distributor of lawn and garden products, whom replaced a 15 year old stand-alone, laptop-based outside sales application with a native mobile app solution using LongRange.

"We needed our sales team to be able to take orders on site, provide customers with order confirmation details, look up open orders and be able to discuss outstanding balances with their customers, and that wasn't happening with our old solution," said Henry Munson, BWI manager information services. "So far, the native app we built for our sales team has been installed on 135 of our US reps' Apple and Android devices. The acceptance rate has been through the roof, our sales team now has instant access to all the IBM i customer information they need while on site, and we've already achieved a 400% ROI in less than a year."

"With the ability for LongRange to run natively on today's Windows 8.1 devices, like the Microsoft Surface and Surface Pro," says Gapp, "companies like BWI can now provide field workers with a single device option that delivers the convenience of a tablet and power of a laptop, there's no more lugging around both while on the road."

CLICK TO ENLARGE

Building on the foundation of LongRange, IBM i programmers don't need to acquire additional skills like HTML5, CSS3, PHP, JavaScript, .Net, Java, or Objective-C to build native mobile apps for Apple, Android and Windows devices. Having the ability to deploy the same application across all three major mobile platforms makes developers productive immediately.



CLICK TO ENLARGE

Key Features of applications built with LongRange

CLICK TO ENLARGE

LongRange consists of server software, LongRange Studio and the LongRange app, downloadable free-of-charge from the appropriate App Store. Programmers use LongRange Studio with RPG, COBOL, LANSA, or CL to build and maintain business-focused mobile apps for the IBM i.

LongRange applications can call programs on IBM i servers, interact with data queues, access DB2 data, etc. From a programmer's perspective, it's just like coding any other IBM i program. LongRange developed applications are easy to maintain and programmers can change or add programs without disturbing the whole application.

Multiple deployment options - Build your apps once and then deploy them, unchanged, to any iOS/Android/Windows device including Windows desktops and laptops.

Mobile device features - Applications can use the camera to take pictures and scan bar codes, interact with Bluetooth connected devices, leverage location services to get GPS co-ordinates and capture information from sensors including the gyro, accelerometer, proximity and ambient light.

Operate without a server connection - Users can continue to operate while offline. Applications can use a SQL database on the device to store data for online and/or offline use, and read and write data to the device's local file system. Run connected, stand-alone, or both.

Platform specific user interface - Since LongRange builds native mobile apps, they inherit the specific aesthetics and behaviour on each platform.

Send and retrieve files - Applications can send files (including documents, photos, and spreadsheets), to a server and retrieve files from a server's file system.

Automated deployment - When a device connects to the server, LongRange deploys changes to the applications automatically.

Security - LongRange supports reverse proxy, SSL and Transport Layer Security (TLS) protocols, IBM i security and authentication mechanisms (up to security level 50), encrypted user credentials and log-in from specific IP addresses.

Choice of Servers - RPG, COBOL and CL-based LongRange applications can be deployed to IBM i servers. LANSA-based LongRange applications can be deployed to IBM i or Windows servers.

Uploading large images with LongRange

A very useful feature of LongRange for LANSA (http://www.lansa.com/products/longrange/) is to give users the ability to upload images from mobile devices to a server via LANSA for the Web.

However, with the ever increasing image sizes (and therefore file sizes) you may incur an issue with LANSA for the Web not being able to upload the larger images.

There is an easy solution for this that can be changed via the LANSA Web Administrator.

1. From the Visual LANSA Settings folder, open the Web Administrator.
2. From the Options menu go to Local Configuration and Select the configuration that is used by LANSA for the Web.
3. Go to the Tools Menu and select Web Server...
4. Now, change the Post Data Size limit to something that is around the file size of the image files produced by your device. For example an iPad may be produce an image of 13 MB so the size limit should be changed to 14MB.
5. Save the changes, exit and retest you image uploading functionality.

A note of caution regarding the Post Data Size limit. This value should not be set to a huge value as a 'just in case' measure as this may introduce unexpected behaviour. Setting this value just above your largest image size is a better approach.

A bug fix for IBM i Access ODBC driver

IBM has released a Service Pack, **SI53809**, containing a bug fix we reported for slow ODBC connection. If you (or customers) develop transformation maps against remote iSeries, you should install the SP.

IBM i Access ODBC driver has a default block size of 256KB, however it was actually blocked by 4KB. The slowness caused by small block size is noticeable when MapForce is querying against remote iSeries. This issue has been corrected in a SP for V7R1. I tested a shipped EDI map against a remote iSeries. It took 89 second to open without the SP; while 23 seconds with the SP.

To install the service pack for IBM i Access for Windows:

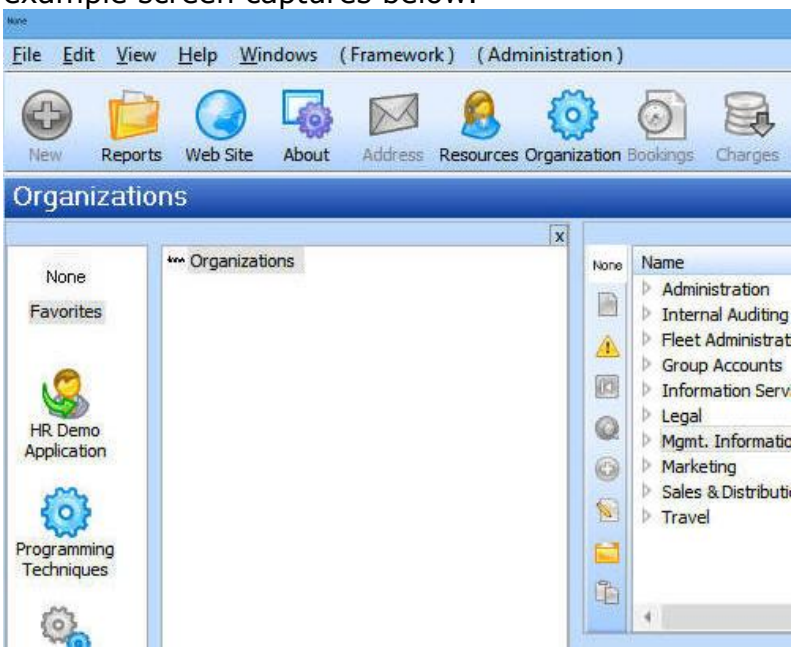
- If you have 7.1 installed on your PC:
  - Download SI53809 package from IBM i Access for Windows Service Pack Web site, and install it
- If you have 6.1 or older (i.e. 6.1 or 5.4) installed on your PC:
  - Uninstall the current version
  - Install 7.1 from software DVD or IFS, and then apply SI53809 if your 7.1 installation image is not the latest level.

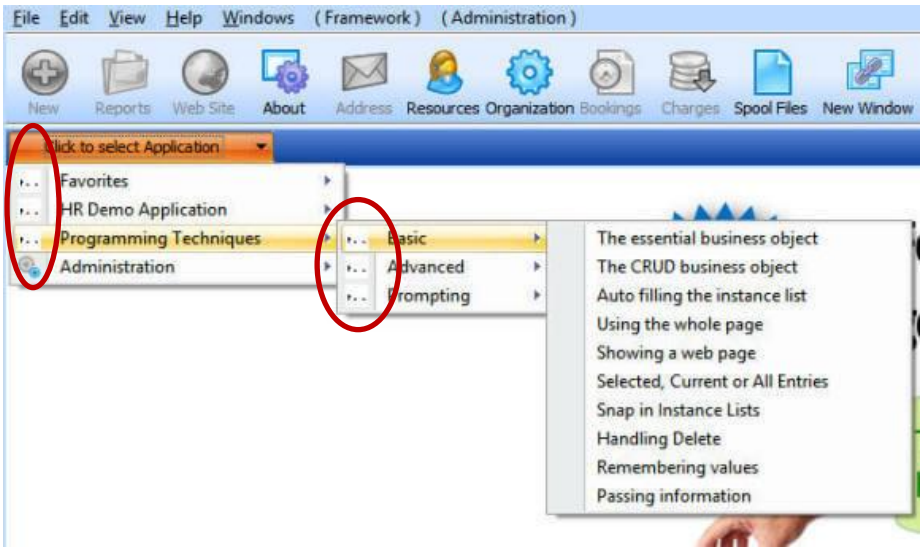
Note

IBM I Access for Windows version is backward compatible. You can use 7.1 Access to connect 5.4, 6.1, 7.1 or 7.2 server.

Blank Icon displays text 'None' in VLF EPC132100

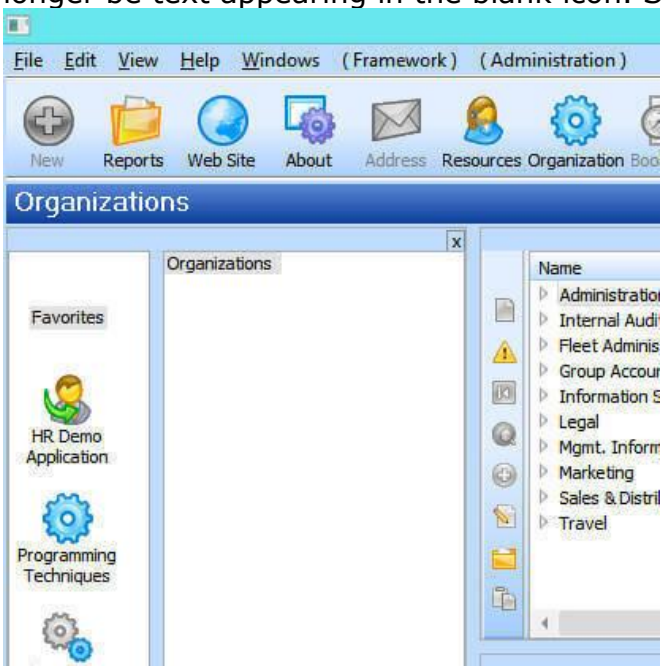
In Visual LANSA Framework (VLF), you have the ability to customise your Framework with both user registered and shipped icons. These icons can be displayed for Applications Views, Business Objects, Command Handlers etc. One useful icon is the blank icon which, as the name suggests, is a blank icon. This can be utilised when no icon is needed for that particular part of your application.
In VLF at EPC132100 level, it was observed that rather than the blank icon being displayed as if there was no icon, the text 'None' would appear in the centre. See example screen captures below.

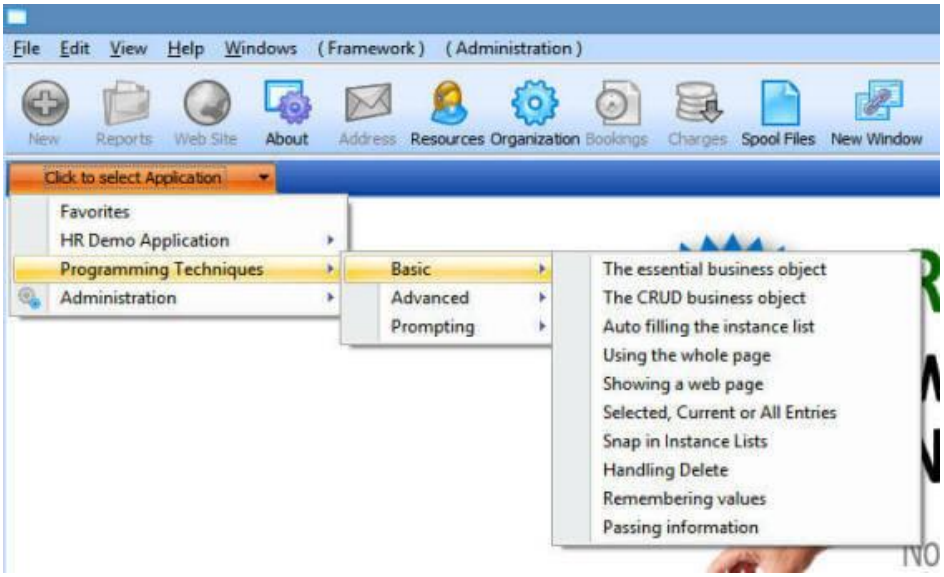




Solution

A hotfix has been created that resolves this issue.
After this hotfix has been applied, the blank icon will be blank again and there will no longer be text appearing in the blank icon. See examples below:





Note

To obtain the hotfix of for any further information, you should contact your local LANSA support group.
This applies to VLF V13 SP2 at EPC132100 only.

Error message that the 'Operating System of this computer is an unsupported platform' at the start of Visual LANSA installation

A Visual LANSA installation will run a series of checks before opening the Installation menu, primarily to confirm that the current configuration of your PC is at a suitable level to be running Visual LANSA.

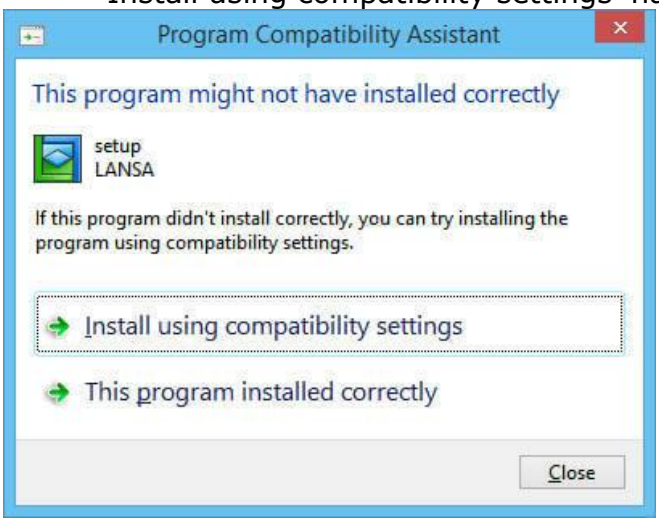
One check is that your Operating System (OS) is supported. If this check fails you will see the following message:

It has been reported that this error can be produced on a version of Windows OS that is a supported version.

So why can this happen on a PC which is at an up to date version of Windows?

LANSA has not fully established why this can sometimes occur but one situation is:

1. a previous installation attempt had been closed or cancelled unexpectedly
2. even if you cancel on the main installation menu before the installation process has taken place
3. a Windows Program Compatibility pop up has appeared.
4. On this pop up rather than selecting 'Close' or 'This program installed correctly' 'Install using compatibility settings' has been selected.



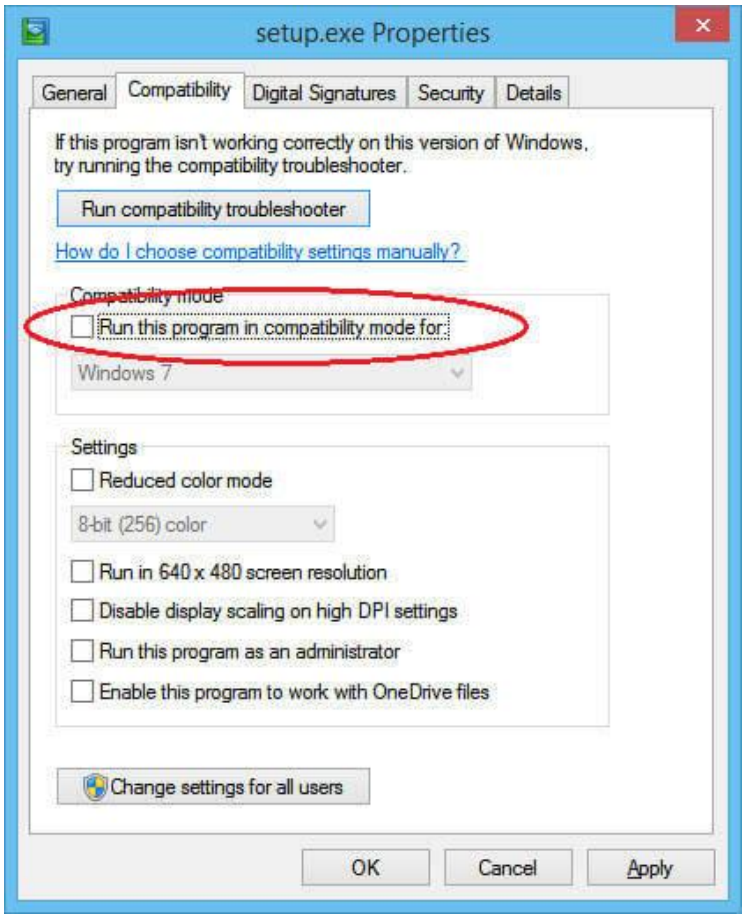
What this setting does it put an entry into regedit that indicates that you run this program in the compatibility of an older version. So, when it comes to the check of the operating system whilst the new install is loading, it will fail because at that point as far as the installer knows it is not a supported operating system.

Solution

There are two solutions for this situation.

Solution 1

1. Open Windows Explorer and navigate to the Visual LANSA DVD. Find the setup.exe that you will be installing from.
2. Right click and select properties
3. select the Compatibility tab
4. Make sure that the 'Run this program in compatibility mode for:' check box is unchecked.



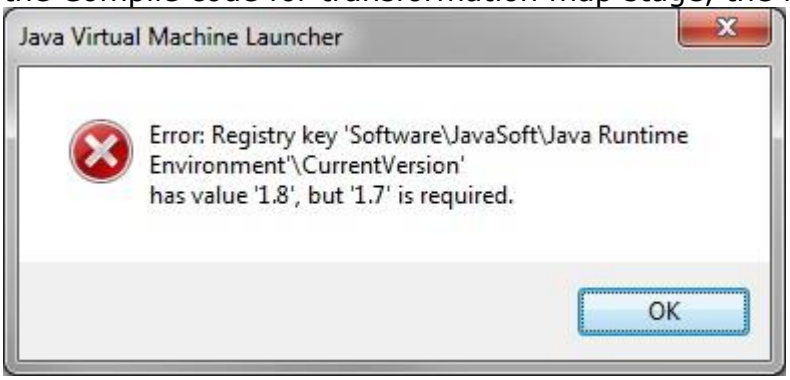
Solution 2

If solution 1 does not resolve the situation and allow the install to start without the 'Operating System of this computer is an unsupported platform' message, you should contact your local LANSA support group for alternative instructions.

Transformation Map prepare error after installing Java 8

A LANSA Composer client installation using an earlier version of Java (for example, the JDK 1.7 that is shipped with LANSA Composer version 5.0) is able to successfully prepare Transformation Maps.

After installing Java 8, Transformation Maps fail to prepare. While LANSA Composer is at the Compile code for transformation map stage, the following error is seen:



After clicking OK, further error dialogs are shown with messages similar to the following:
Error: could not find java.dll
Error: could not find Java SE Runtime Environment

Cause

LANSA Composer's map prepare starts Java to compile the Java code using an unqualified reference to javaw.exe. That is, it relies on the usual Windows search algorithm to locate the executable file. The result is that it first searches pre-defined Windows locations (such as C:\Windows\SysWOW64) and then locations specified by the PATH environment variable.

When the earlier Java version was installed it installed copies of one or more of the Java executables java.exe, javaw.exe and/or javaws.exe in one of the following locations (depending on whether the OS is 32-bit or 64-bit):
C:\Windows\System32
C:\Windows\SysWOW64

When Java 8 is installed, it installs shortcuts to its
own java.exe, javaw.exe and/or javaws.exe in a new location, which it also adds to the start of the PATH environment variable.
C:\ProgramData\Oracle\Java\javapath

It does not update, replace or remove the existing executables files in C:\Windows\System32 and/or C:\Windows\SysWOW64.

(Note that it might only be necessary that the Java 8 JRE (not necessarily the JDK) was installed to lead to this issue. For example, installing a Java application that install Java 8 JRE might also lead to the same result.)

When LANSA Composer attempts to start Java using javaw.exe, the Java 7 executable in C:\Windows\SysWOW64 is started. As javaw.exe proceeds to load further executable files it requires (e.g.: java.dll), something in the new configuration causes it to locate and load the Java 8 executables, which are incompatible with the earlier version, leading to the reported errors.

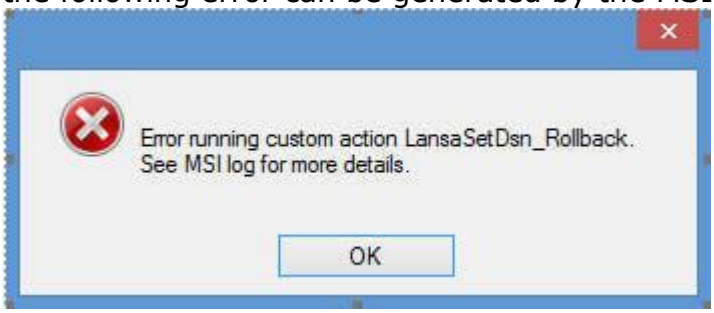
Suggested Resolution

1. Remove the "old" Java executables from C:\Windows\System32 and/or C:\Windows\SysWOW64. In our testing, this step resolved the issue and allowed the Transformation Map prepare to proceed to normal completion.
2. Alternatively, try changing the PATH environment variable value to put the Java 7 location ahead of the new Java 8 location.
3. Temporarily change the registry entry (Software\JavaSoft\Java Runtime Environment\CurrentVersion) from 1.8 to 1.7 before compiling maps. After the compiles, replace the registry value, if required.

Of the 3 suggested solutions, in our testing, suggestion 1 was the only one that gave consistent results (allowing the transformation map prepare to succeed). Other OS factors, beyond the scope of our testing, may be at play in suggestions 2 and 3.

MSI Installer: Cannot uninstall deployed application

When trying to uninstall a V13 Visual LANSA deployed application, in a certain situation, the following error can be generated by the MSI Installer.



This can happen if the following conditions are met:

- You have chosen to deploy or use an INI file with your application (as a non-LANSA object)

OR

- Your application creates an INI file (using TRANSFORM_LIST, or Stream File Built-in Functions, or using DOS commands etc.)
  and
- The name of this INI file is the same as the application name defined in the Deployment tool
  and
- The location of this <appname>.INI file is the System Execute directory

In either of these situations, the deployment tool will pick up this INI file during the version/patch install or uninstall and use the first 2 lines to overwrite the DBUT and DBII values in your application. This is an internal mechanism that is intended to be used for Network Client application setup, but if you have your own INI file then the values in those lines may cause problems.

Solution

To avoid getting into this situation, you may choose any one of the following workarounds.

1. Change the name of the INI file [to not be the same as the application name]
2. Change the name of the application [to not be the same as the INI file]
3. Alternatively, if changing either names is not an option, then put the file in a different location. As a suggestion, you can use the partition execute directory or the system root directory.

If you are already encountering this error during uninstall of a V13 deployed application, you should contact your local LANSA support group for recovery instructions.

Digitally signing LANSA application DLLs

Digitally signing LANSA application DLLs (i.e. DLLs for Forms, Reusable Parts, Functions, IO Modules etc.) may be required for audit purposes, and can be used for whitelisting by antivirus providers. LANSA provides a feature for digitally signing deployment tool packages, however the digital signing of LANSA objects is not currently possible from within LANSA.

The reason for this is that it may not be appropriate for developers to be signing their DLLs every time they compile an object. The signing of DLLs should only be done as part of the final release process, much like creating a deployment package. As such it could be implemented using some sort of script that the release manager employs. This should be written/designed yourself and depends on the scripting language/signing tool being used.

A typical build/release process might be:

1. Compile all DLLs
2. Sign all DLLs using a script
3. Package all DLLs for inclusion in an MSI version

For a patch release, only the required objects would be compiled and signed before packaging up.

As a final note it is important to ensure that the certificate is handled securely, by adding it to the key store on the build machine and password protected so it can only be used on that build machine and cannot be exported.

Can host monitor files be omitted from backing up before upgrading LANSA for i?

Host monitor files store the connection details of each bit of work of the Visual LANSA slaves that connect to an IBM i master. These files are contained in subdirectories under the x_hmrqst directory on the IBM i IFS.

When upgrading LANSA for i, it is a vital step to back up your system before commencing so that it is easy to restore if any issues are faced.

While backing up is an essential task, it can be time consuming when there are large quantities of directories and files to be backed up.

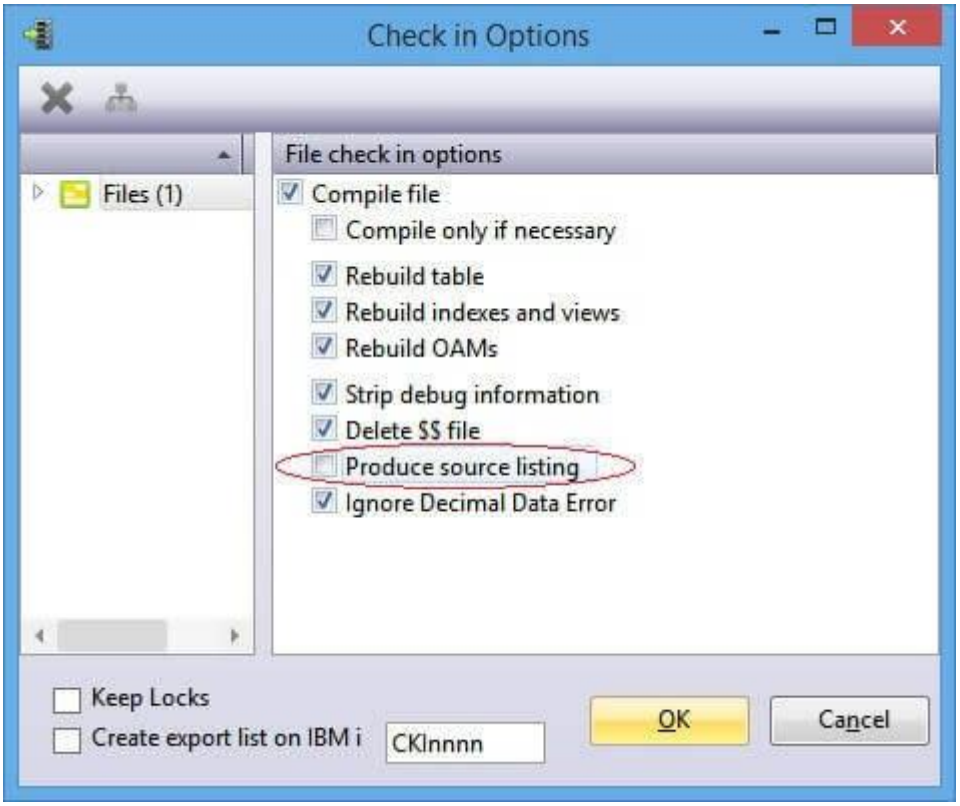
Can x_hmrqst be omitted from your back up strategy, either via your daily/weekly back up or before upgrading LANSA for i?

Firstly it is important to understand what is contained in the x_hmrqst directory. It contains subdirectories that reflect the Visual LANSA slaves that have been connected to a LANSA for i master system. These subdirectories contain files that relate to the connection information between a LANSA for i system and the Visual LANSA slaves.

These files are refreshed every time the host monitor is started in Visual LANSA. If they do not exist they will be created automatically. This means that for systems that have many developers working on a single system the x_hmrqstdirectory can quickly fill up with many subdirectories, containing many files, which in turn can potentially slow down the back up process.

In principle, yes, the x_hmrqst directory can be omited when backing up your LANSA for i system before upgrading. As long as you are not compiling objects out side of LANSA then there is not immediate need to keep a backed up version for an upgrade.

Furthermore, to prevent these sub directories from accumulating too many files in the first place uncheck the 'Produce Source Listing' checkbox when compiling and checking in objects. There is no reason to check this option if you are not compiling outside of LANSA.



Note

Restoring from back up is an important step if any upgrade issues occur. If the time for the back up to run is not an issue, then including the x_hmrqst directory is advisable and should only be omitted if it is substantially slowing the time it takes to back up your system.

Understanding how DateTime fields work in LANSA

The use of DateTime fields in LANSA without understanding how they work can lead to unexpected results at runtime. The usual complaint is that the time portion is showing several hours later/earlier than it should. But once you understand how they work and how to code with them, you will have an application that is open to use from around the world, no matter where your users are, or whether they use daylight savings or not.

DateTime field type explained

The following document aims to explain DateTime fields and how they are used in LANSA: Getting started with DateTime (PDF 137KB) (http://www.lansa.com/downloads/support/datetime.pdf)

Further notes

As per the document, DateTimes are processed as UTC (Greenwich Mean Time) values in LANSA. This means that DateTimes created programmatically (through literal values, or as output from Intrinsic functions such as AsDateTime) are assumed to be the time at UTC.

For example, if I set a date as follows:
#StartDate := := ('20150320123456').AsDateTime(CCYYMMDDHHMMSS)
or
#STD_DTIMX := '2015-03-20 12:34:56'

Each of these statements will produce a date as 12:34:56pm, 20th March 2015. But this is assumed to be at UTC time (i.e. in Greenwich in the UK). When it is 12:34:56pm in the UK, it is actually 23:34:56pm in Sydney, Australia (UTC +11hrs in summer time). So if my #StartDate field has the DUTC attribute disabled (default), when I look at this field on my form/WAM, it will show 11:34pm, despite me hardcoding the time as 12:34pm in my application.

The same situation will also occur if you piece together a DateTime from two separate Date and Time fields.

It all seems too difficult!

If all you do is accept DateTime input from the customer, store those in a database, and later retrieve and display that information, then you do not have to think about timezones and UTC. Whenever a date is input or displayed or stored in the database, the UTC conversion is done automatically by LANSA, so they will see whatever time that they entered.

If you need to hardcode date/times, or create DateTimes from separate dates and times, then the following intrinsic function is all you need to remember:
.AsUniversalDateTime()

This intrinsic will subtract the UTC offset from your generated date, to produce the time at UTC for when the local date/time is that value which you are generating.

For example, both of these statements will now display 12:34:56pm at runtime:
#StartDate := ('20150320123456').AsDateTime(CCYYMMDDHHMMSS).AsUniversalDateTime()
or
#StartDate := ('2015-03-20 12:34:56').AsUniversalDateTime()

Alternatively, if your application will never be used outside of your current timezone, you may consider enabling the DUTC flag for DateTime fields as explained in the document. In that case all DateTimes, displayed on screen, or generated in code will be at the same time offset (UTC, or +0).

The 'Enable for Long names' option has moved

Some users have questioned why the 'Enable for Long Names' partition option is no longer available in V13 SP2. In fact, this option is still available but was removed as a partition setting in V13 SP1.

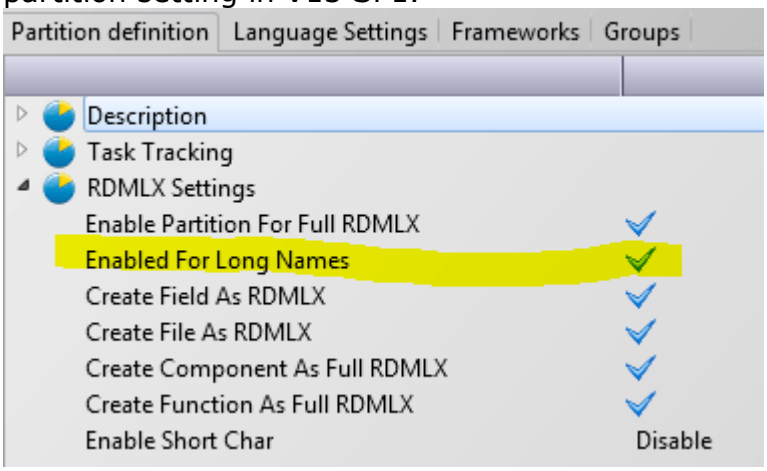


Figure 1 shows V13 Partition definitions.

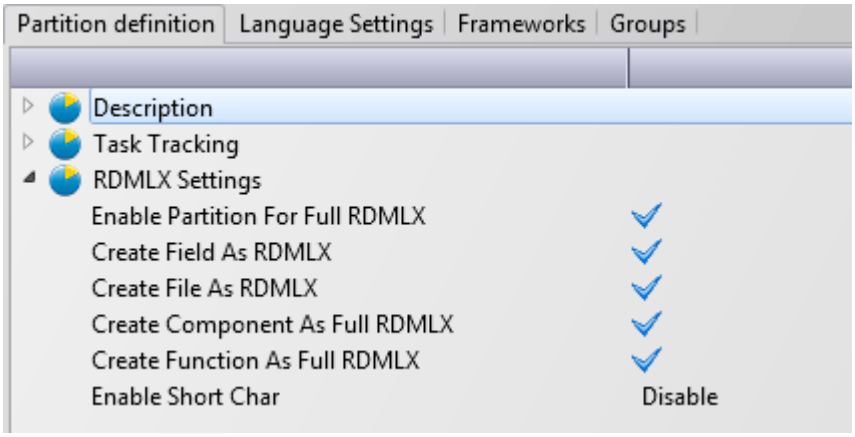
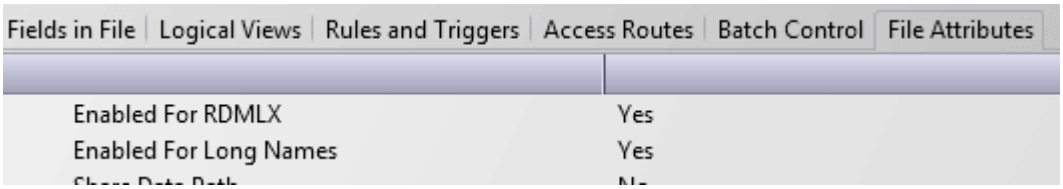


Figure 2 shows V13 SP2 partition definitions, with Enable for Long Names no longer listed.

Initially, 'Enable for Long Names' was made a partition option so users could understand the impacts and consequences of using long names. For example, in a situation where a user could give fields long names, recompile their files and the column names would change and any non-LANSA code using these tables might be affected.

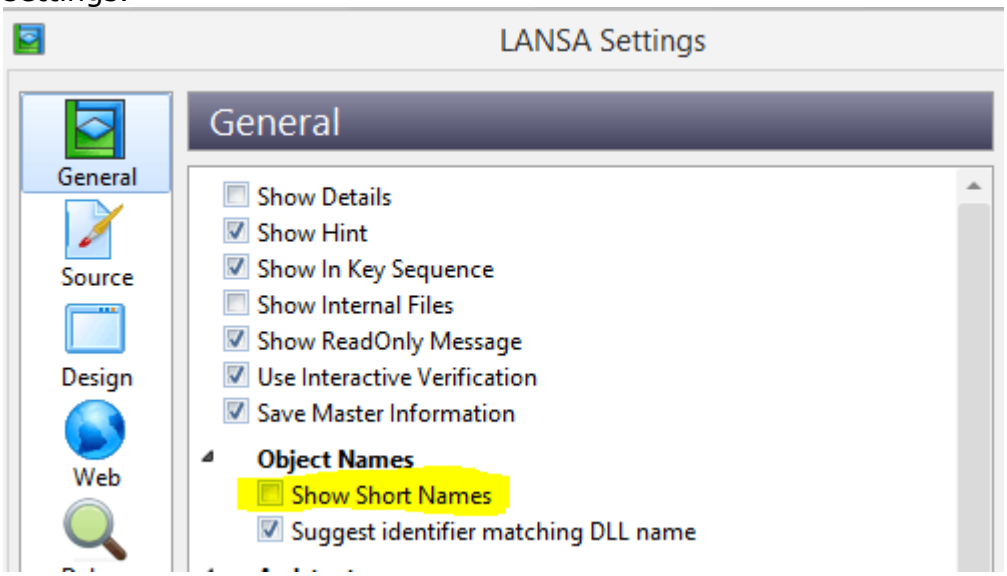
However, given that you are always enabled for long names in an RDMLX partition, a partition level option is not required. Instead, since V13 SP1, the option is available as a File attribute via the 'Enable for Long Names' flag on the individual file.



Note

There are still some references in the Online documentation to the partition level option. These references will be removed.

If you wish to only View short names in the IDE, this can be set from the General LANSA settings.



Guidelines for upgrading Visual LANSA including the SQL Server Instance

Users that currently have Visual LANSA V12 SP1, with the shipped SQL Server Express 2008 installed, have 2 migration paths available to get to V13 SP2, which ships with SQL Server Express 2012.

The migration is slightly different depending on whether users wish to

- upgrade their current Visual LANSA V12 SP1 to V13 SP2
- or whether users wish to install a separate V13 SP2 and also keep the existing V12 SP1 Visual LANSA environment (side by side).

If you wish to upgrade from Visual LANSA V12 SP1 (using SQL Server Express 2008) to V13 SP2 (using SQL Server Express 2012), you must upgrade your SQL Server 2008 instance to a 2012 instance prior to upgrading Visual LANSA.

Use the steps in the attached document to complete the upgrade:
Guidelines for installing both V12 SP1 and V13 SP2 development systems using a single SQL Server 2012 Express instance
(http://www.lansa.com/downloads/support/Upgrading-a-dev-system-from-V12-with-SSEE2008-to-V13SP2-with-SSEE2012.pdf)

If you wish to install a new Visual LANSA V13 SP2 (using SQL Server Express 2012) and also maintain your current V12 SP1 (currently using SQL Server Express 2008), you must upgrade your SQL Server 2008 instance to a 2012 instance prior to the V13 SP2 Visual LANSA install.

Use the steps in the attached document to complete the upgrade:
Guidelines for upgrading a development system from V12 with SQL Server 2008 Express to V13 SP2 with SQL Server 2012 Express
(http://www.lansa.com/downloads/support/Installing-both-V12SP1-and-V13SP2-dev-systems-using-a-single-SSEE2012-instance.pdf)

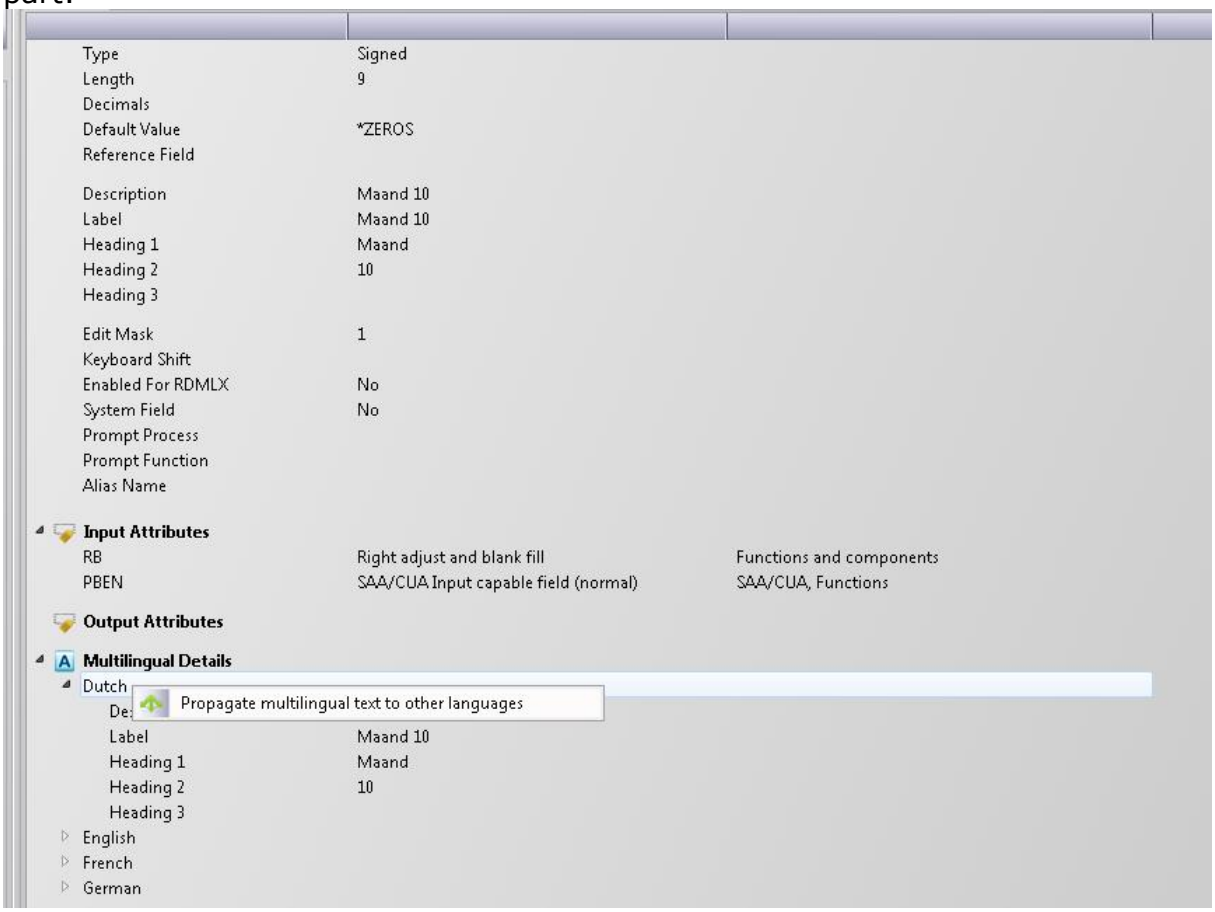
Using Fiddler for Internet troubleshooting

Fiddler runs on your PC, monitoring all traffic between your web browser and the internet. LANSA Technical Support may ask you to follow the instructions below to produce a Fiddler trace file as part of the problem resolution.

Does it have to be installed?	Yes. From http://www.fiddler2.com/fiddler2/version.asp
How you turn it on?	Run the Fiddler application on the same PC as the browser. To start capturing traffic select Capture Traffic from the File menu.
How you turn stop it?	Deselect Capture Traffic from the File menu or close the Fiddler application.
How is it typically used to isolate problems?	Open your web browser and clear your browser cache Close your web browser. Start Fiddler and turn on the Capture Traffic option. Reproduce the problem in your web browser. In Fiddler - select File > Save -> All Sessions Send the resulting .saz (Session Archive) file to support.

Did you know?

That when you use your right mouse button at a language in the Multilingual Details part:



That you can 'Propagate multilingual text to other languages'?