LANSA Application Deployment Tool

- About this Guide
- Introduction
- What's New?
- Plan your Deployment
- Getting Started with the Deployment Tool
- Configure the Deployment Tool
- Applications
- Versions and Patches
- Options & Settings
- Deploy LANSA Communications
- Deploy Client / Server Applications
- Deploy LANSA for the Web Applications
- Just In Time Upgrade Guidelines
- Deployment Templates
- Translate a LANSA-developed Application
- Review your Deployment Package
- Test your Deployment Package
- Install an Application
- Upgrade an Existing MSI Application
- Upgrade an Existing V12 Application
- Cloud Customizations
- How to...?
- Deployment Tool Tutorials
- Cloud Tutorials
- Glossary

Edition Date June 15, 2015

© LANSA

About this Guide

The LANSA Application Deployment Tool is used to deploy your finished applications from the Visual LANSA development system to a runtime environment on a PC.

This guide describes the many options and settings required to deploy a Visual LANSA application onto a target PC.

It is intended for use by developers or other staff experienced in the use of Visual LANSA and with knowledge of Windows operating systems.

You are also advised to become familiar with the terminology used in this guide. For this, refer to the Glossary.

The text in this guide is available while using the Deployment Tool by selecting the *Contents* command from the *Help* menu on the Deployment Tool's main window or by pressing F1 to invoke context help while using the software.

Before you use this guide, you should review the following:

Who should use this Guide

How to use this Guide

Additional Information.

For the latest product information and updates, please refer to the technical information section of the LANSA Web site at www.LANSA.com/support

Always use the documentation relating to your version of Visual LANSA.

If you are deploying an IBM i application, refer to the guide *Deploying LANSA Applications on IBM i*.

Who should use this Guide

This guide has been written for Windows developers or administrators who are responsible for deploying applications on to customers' PCs - whether the customers are internal or external to the organization.

It is assumed that you have some knowledge of Visual LANSA, as well as the target PCs characteristics and requirements before beginning to create a deployment package.

Detailed knowledge of the LANSA application to be deployed is also required, as all the objects used by the application must be selected for inclusion in the package - or in one or more packages.

How to use this Guide

This guide contains detailed instructions for the use of the LANSA Application Deployment Tool. It has been divided into sections for your convenience:

- For a general overview and notes to review before you start creating packages, refer to the Introduction.
- Applications describes the Applications into which you group your Versions and Patches for deployment.
- Package Options & Settings describe all the options that you could specify to define the details of the packages you wish to deploy.
- Templates can be used to provide a basic structure for a version based on common deployment scenarios.
- Translating a LANSA-developed Application details how to translate the LANSA application objects and incorporate them into the final application.
- Versions and Patches covers the tasks involved in creating and delivering packages.

If you are upgrading from an earlier version of the Deployment Tool, be sure to check Upgrading Existing V12 Application before you start.

Additional Information

For more details about installing, configuring or troubleshooting LANSA software, you may need to refer to these guides:

- Installing LANSA on Windows
- LANSA Communications Setup

Both of these guides are supplied with the LANSA online guides on the LANSA Windows DVD.

For the latest product information, refer to the LANSA product Web site at www.LANSA.com/support

1. Introduction

For an overview of the LANSA Deployment Tool, please review:

- 1.1 What is the Deployment Tool?
- 1.4 Who should use the LANSA Deployment Tool?

Deployment Quick Tour in Getting Started with the Deployment Tool

1.1 What is the Deployment Tool?

The Deployment Tool enables you to package and distribute your Visual LANSA-developed product to your customers, whether they are internally or externally based.

Using the Deployment Tool, you start by creating an Application. Within the Application you create a Version containing the relevant objects that make up your Visual LANSA-developed product. You select specific LANSA objects for inclusion in the Version as well as nominating options and settings that relate to the environment in which the Version will be deployed. Non-LANSA objects can also be included in a Version.

Once you have completed the selection process the Version is built which produces an MSI file.

Your Visual LANSA-developed products are thus distributed to the end user for the installation of all the relevant objects on to their PC. All the user has to do is run the MSI file and follow the prompts - or you can use the Just In Time Upgrade to automatically update the user's version of the application, as necessary, each time the software product is started. (See Glossary for a description of the MSI file and Just In Time Upgrade.)

Your software product can be distributed and deployed using:

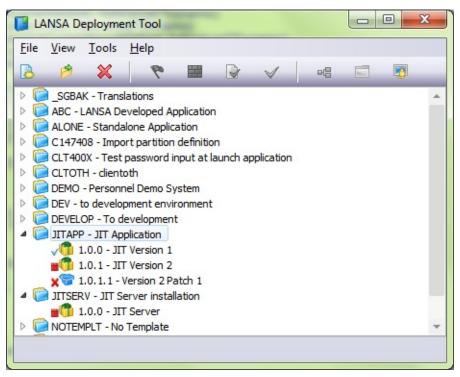
- An automatic installation from an application serving system
- A web based download which is initiated by operator selection
- A copy of the MSI file on to media such as a DVD
- Systems Center Configuration Manager software.

Applications may be maintained by creating a Version or a Patch based on an existing Version. A new Version may be used for a first installation of the software product or may be used to upgrade an existing installation of the software product. A Patch must be applied to an existing installation of the Version on which it is based.

1.2 The Deployment Model

A Deployment Tool Application in combination with the first Version created within the Application defines the core of your software product and becomes the building block for subsequent Versions and Patches in the Application.

The software product definition is contained in the Version and Patch details when the Version or Patch is built, and the resulting MSI file (Version) or MSP file (Patch) is deployed.



Also see

- 1.2.1 What is an Application?
- 1.2.2 What is a Version?
- 1.2.3 What is a Patch?
- 1.2.4 What is a Package?
- 1.2.5 What do you need to deploy?

1.2.1 What is an Application?

The concept of an Application important and binding. In previous releases of LANSA, an Application could consist of a collection of unrelated packages, or packages used for different purposes such as a client install package and a server install package. A LANSA Application is synonymous with a software product.

Now, the first Version in an Application defines the full product, including all objects and settings, and is used as the basis for subsequent Versions and Patches of the Application. Install options are used to determine what and how the package is installed, for example if installing with database or installing client portion only.

In the Deployment Tool interface the *Application Name* identifies an Application. It has an Application description and Company Name, which are visible to the end user and are used when installing and uninstalling the software product.

However, from a Windows Installer perspective, an Application must have a unique identifier that relates all of the Versions and Patches within this Application to each other. Each time a Version or a Patch is built, another unique identifier is generated for this particular build. These are Globally Unique Identifiers (GUIDs) and are stored at the Application level. The MSI and MSP installations internally use these unique identifiers to validate the installs. For example, when the second Version of a software product is installed the Application's unique identifier is used to see if a previous Version is already installed. If it is already installed the Version will be automatically uninstalled before installing the second Version of the product.

1.2.2 What is a Version?

A Version is a full install of the software product. A Version belongs to an Application.

A Version is identified by its Version Number. The Version Numbering is arbitrary but subsequent version numbers must be greater than previous ones. For example, 1.0.0 then 1.0.1 then 1.3.0 then 1.3.3 and so on. Notice that numbers can be skipped but the sequence must be increasing.

A suggested way to use your Version numbering is:

<major release>.<minor release>.<build number>

A Version may include:

- LANSA objects
- Non-LANSA objects
- Database setup information
- SuperServer setup information
- Communications details
- Just-in-Time delivery information
- Shortcuts.

Whatever your software product requires to execute and run correctly must be included in the Version.

If the Version includes the setting *Deploy to Development System*, no MSI file is produced. The internal object definitions must be imported from the internal files. Refer to LANSA Import for further information.

1.2.3 What is a Patch?

A Patch is used to deliver a set of changes to a software product that has been installed using the Windows Installer. A software product can be upgraded by installing a new Version (MSI file) or by applying a Patch (MSP file).

- A Patch is typically used to apply modified compiled objects, shortcuts and executables.
 - You should NOT use a patch to deploy database changes as they are complex to manage and can easily result in a corrupted database.
- A Patch is identified by its Patch Number.
 The Patch number has a direct relationship to the Version or Patch it was based on, for example Patch 1.0.0.1 would be the first patch against Version 1.0.0.
- To create a Patch, copy a parent Version or Patch.
 In this way the Patch inherits the parent definition including the objects to be deployed.
- When the Patch is built, a full MSI file is built in exactly the same manner as the parent Version or Patch was built.
 The resulting MSI file is compared with the Version's MSI file to produce a difference file – the MSP file.
- Note that only the MSP file is installable.
 The interim MSI is deleted once the MSP has been constructed.

The MSP file is a self-contained package that contains updates to the software product and describes which Version can receive this Patch. In many cases it is advisable to service products by delivering a Patch because an MSP file is typically much smaller.

If a patch is only to be used to ship updated objects and there are no new objects, then nothing needs to be changed in the Package Definition. Just save the Package and build it. Any differences between the Version and the current state of objects will be included in the Patch.

A Patch is usually created from the last Patch definition, so that it includes any changes made to the Package Definition, like the addition of new objects.

The order of the application of Patches does not change the installed result. If Patch 1 contains MYFORM.DLL version 1.0.0.1 and Patch 2 contains MYFORM.DLL version 1.0.0.2 then whether Patch 1 is installed first or Patch 2 the result is that MYFORM.DLL version 1.0.0.2 is installed.

Similarly when a Patch is uninstalled. If Patch 1 is now uninstalled, it will only affect the version of DLLs which it installed. As version 1.0.0.2 is installed, uninstalling Patch 1 will not replace MYFORM.DLL with version 1.0.0.0. It will still be version 1.0.0.2. Only when Patch 2 is then uninstalled will it be restored to version 1.0.0.0.

This may sound neat, but it can become complex to manage if Patches are created that are not cumulative. Each Patch should include all the changes that have come before, or back them out if necessary, and be considered as the ONLY patch required to move the installation to that Patch level. So no matter what Patch level a user has chosen to install, to get to the latest version they just need to install the latest Patch. No interim Patches are required.

Note: By creating patches sequentially from a single build machine and not restoring any objects to a prior state will ensure that Patches are cumulative.

Also see

1.2.2 What is a Version? for information on versioning.

1.2.4 What is a Package?

The terminology Package is still used as a generic reference to a Version or Patch.

1.2.5 What do you need to deploy?

Deploy the MSI file or MSP file resulting from the build of your Version or Patch respectively.

If a Version or Patch includes the setting *Deploy to Development System* (specified in Settings) the build processing will not generate MSI or MSP files. In this case the definitions must be imported from the internal file definitions. Refer to LANSA Import for further information.

1.3 Upgrading a LANSA-developed Application previously deployed

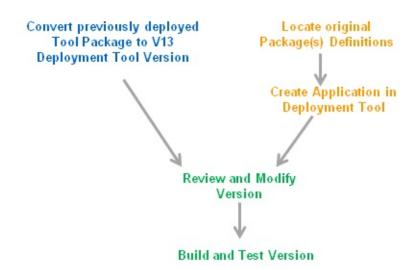
If you have an existing Application which needs to be modified after upgrading your development environment to LANSA V13 you will need to:

Formulate a new deployment plan. Refer to 1.2 The Deployment Model.

Create an initial Version to install the complete application using Microsoft Windows Installer

Remove references to the old Application from desktop. Refer to Upgrade Options.

There are two different methods which can be used to upgrade a LANSA-developed application which was deployed using an earlier version of the Deployment Tool (pre-version 13).



Step How to do it

1. Convert Package to Version

Identify the base package to be used as a starting point for your new Application. You may not have all the details required in a single package so it is recommended to select the package which includes the main set of application objects (files, forms etc...) to be deployed. The package setting can be modified as required after conversion. Refer to Convert a Pre-V13 Package.

1a. Locate Locate the original package(s) used to deploy your Application.

original package definitions In this scenario you will not be copying the package but using the information to construct a Version definition.

1b. Create in

Create an Application and Version in the Deployment Tool. Select Application an appropriate template to assist with the construction of your Application.

Deployment Tool

2. Review

Version

Review the Version and ensure all required settings and objects and Modify have been included. Ensure ALL objects and settings are provided whether they relate to client, server, web-server or database.

3. Build and Build the Version and install into a clean environment, or into a test Version test environment where previous package was installed (to test the upgrade options).

1.4 Who should use the LANSA Deployment Tool?

There are many uses for the LANSA Deployment Tool and these will become apparent as you get to know how the tool works. It is specifically designed to satisfy the needs of:

- Organizations with software to be distributed to internal or external customers
- Developers using Visual LANSA.

2. What's New?

Following is an outline of the new features in this version of the Deployment Tool.

- 2.1 A New Deployment Model
- 2.2 File Library Automatically set to Partition Library
- 2.3 Set File Library
- 2.4 Control File Data Loading and Unloading
- 2.5 Network Client Installation decided during Installation
- 2.6 Runtime always Included in Installation Files
- 2.7 Translation Tool integrated into Visual LANSA Tools
- 2.8 Unknown objects are no longer dropped when Package is saved
- 2.9 Host Route Table and Listener defined at Application Level only
- 2.10 Backup or Restore Applications
- 2.11 New option Require Elevation
- 2.12 New options to install Web Features
- 2.13 Options not selected to Prompt during Installation are no longer shown

2.1 A New Deployment Model

Application installation uses Windows Installer and Windows Installer XML instead of InstallShield SETUP.EXE to install your software product.

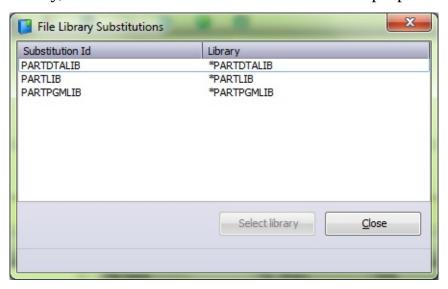
The Deployment Tool interface will look familiar if you have used the Deployment Tool in previous releases of LANSA. The interface has been simplified to some degree, as several options are no longer relevant in the Deployment Model. Refer to The Deployment Model for more information.

2.2 File Library Automatically set to Partition Library

File Library is soft and will be determined during installation based on the environment libraries. By default files will be installed into the Partition Data Library.

2.3 Set File Library

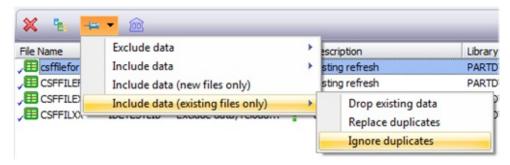
The File Library can now be configured when files are included in a package. During the Package Installation the file is updated and installed with the appropriate library, table collection and owner information properties.



2.4 Control File Data Loading and Unloading

When installing a package that includes files which already exist in the target application, in the package definition you can pre-determine the order in which:

- data is unloaded from the existing file
- new data shipped in the package is loaded into the target file, and
- existing data reloaded.



2.5 Network Client Installation decided during Installation

Network Client in no longer an option associated with the Application definition. Now every installation puts in place a Network Client MSI that may be used to install a Network Client if required.

2.6 Runtime always Included in Installation Files

The options *Deploy LANSA Execution Environment* and *Deploy with EPCs* have been removed. The Microsoft Installer Files will always include the current LANSA Execution Environment and latest EPCs.

Deploy with VC90 Runtime has also been removed as this was a requirement to support Installshield.

2.7 Translation Tool integrated into Visual LANSA Tools

The Translation Tool has been integrated into the Visual LANSA Tools menu. There are two relevant options:



- Translate Object Details is the interface to be used by the translator to view
 the translation files created by a Translation Application. The details relating
 to the objects can be translated, updated and returned to the Application
 builder. Refer to Translating a LANSA-developed Application for more
 information.
- Import Translations allows the Application builder to import translations received from a translator into the current development environment.

2.8 Unknown objects are no longer dropped when Package is saved

Previously, if your package included objects which were not known in the local repository when the package was saved these objects were dropped from the package.

In the current release any unknown objects must be resolved (checked out, imported or created) before the package will build successfully. If the objects are not required they must be manually removed from the package.

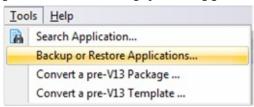
2.9 Host Route Table and Listener defined at Application Level only

Previously, if a Host Route Table and Listen file could be included at either Application or Package level.

In the current release the Host Route Table and Listen file can only be defined at Application level as this information should pertain to all associated Versions and Patches.

2.10 Backup or Restore Applications

Applications are now identified by a set of GUIDs (Globally Unique Identifiers) which are generated when the first Version is successfully built. This set of GUIDs will be used to identify the Application and facilitate upgrades and patches. Without these GUIDs you cannot upgrade or patch your software product so backup your Application definitions regularly including the GUIDs.



2.11 New option Require Elevation

An additional option Require Elevation has been added to determine if the installation requires administrator privileges to proceed with installation.



Execution parameters

 Require Elevation
 No

 Language (LANG)
 [Current]

 Process to Execute (PROC)
 [None]

 Function to Execute (FUNC)
 [None]

 Form to Execute (FORM)
 [None]

 Partition Identifier (PART)
 [Current]

 LANSA User (USER)
 [QPGMR]

2.12 New options to install Web Features

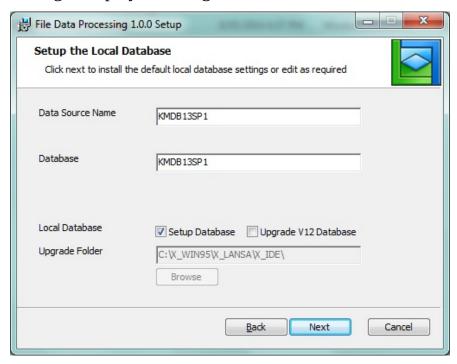
New options have been added to support the installation of Web Administrator, IIS Plugin, Web Images, Java Service Manager and Visual LANSA Framework. Previously these features were required to be installed from a LANSA installation image.



Additional LANSA Features	
Install Listener	No
Install Web Administrator	No
Install IIS Plugin	No
Install Web Images	No
Install Java Service Manager	No

2.13 Options not selected to Prompt during Installation are no longer shown

In previous versions, options which were not selected to Prompt During Installation were shown during the installation as unchangeable values. These values are no longer displayed during the installation.



3. Plan your Deployment

Before you create a Deployment Tool application, consider how you will be best to organize the deployment of your Visual LANSA Application.

☆ It is important to understand that deployment is more complex than "just picking up everything and moving it over there".A successful deployment requires understanding, planning and thorough testing.

Start by considering the following questions:

- **Where** is the application going to be installed?
 - Client? Server? Standalone?
 - Windows? Linux?
- **How** is the application to be installed and subsequent upgrades applied?
 - MSI or MSP file?
 - Per-user or per-machine?
 - Just in Time upgrade?
 - SCCM integration?
 - Into a development environment?
- Do you need to distribute different parts of the application to **different** machines?
- How does the application **connect to the database**?
- Are there any **templates** suitable to use as the basis for you Application?

If you can answer all of these questions, then you are ready to start deploying your application.

If you could not answer all of the questions above, you need to develop a clearer understanding of what you are attempting to deploy before you proceed!

Establish the minimum supported configuration

You need to formally define the minimum configuration your solution will viably need, including:

- Minimum hardware requirements
- Minimum software requirements

- Minimum networking capabilities
- Maximum data volumes.

A formal minimum supported configuration will:

- Allow you to make informed decisions about the overall solution cost
- Establish the environment required to test the deployment of the solution or any patch/hotfix made to it.
- Raise management's awareness of the risk in implementing a sub-minimum supported configuration solution.

Repercussions of Per-Machine vs. Per-User Installations

Windows Installer can install or upgrade a software product for all users of a computer (per-machine c ontext) or for a particular user (per-user context) based on the access privileges of the installing user. If the initial Version is installed per-machine then all subsequent Versions and Patches must be installed per-machine.

If you plan to deliver your software product updates using Just-In-Time upgrade, then the users who is running the application MUST have the rights to install those updates. So if they do not have the rights to install per-machine and the software is installed per-machine then when the update is attempted it will fail.

Also see

- 3.1 Deployment Tool Environment
- 3.2 Lifecycle of an Application
- 3.3 Version or Patch Upgrade?
- 3.4 Per-User or Per-Machine Install?
- 3.5 Digital Signatures
- 3.6 Installation Mode
- 3.7 SCCM Integration

3.1 Deployment Tool Environment

The recommended configuration for creating Deployment Tool Applications is:

Recommended Deployment Tool Environment Deployment applications are copied to an Application Server Deployment Master Application System repository Server A stable version of Application is copied Application to the Deployment development is Target PCs install System checked into a the application master from the repository Application Server Development **PCs** Target PCs

Development Visual LANSA slave systems are installed on the Development **PCs** PCs for the ongoing application development.

A standalone system may be used if there is only one developer.

Master Repository

All development work must be saved to the Master Repository.

Deployment System

It is recommended to have a separate Deployment System, with a different master repository, to ensure on-going development is not inadvertently included in a package.

Application Server

The Application Server is a simple file server and as such does not require a powerful processor.

After a package has been distributed and tested it is ready to install on a Target PC. It is recommended to have a separate Application Server to ensure the package is not inadvertently rebuilt or modified.

If the Application Server is used for Just In Time processing it

will wait for requests for upgrades from a Target PC before installing the packages on the Target PC as required.

If Just in Time processing is not used the package installation on the Target PC must be manually initiated.

Refer also to 3.1.1 Using a Workstation as a Server.

Target PCs

This is where the package will be installed and the application executed.

This includes PCs used for testing the package and installation process.

MS SQL Database License

PCs and **Deployment System**

Development Visual LANSA developers with a Visual LANSA development license are licensed to use the MS SQL database server software. This applies whether Visual LANSA is installed as part of a Network or Standalone installation.

Target PC

A separate license is needed for applications that are deployed using the MS SQL software.

3.1.1 Using a Workstation as a Server

The listener is automatically installed on PCs running Windows **Server**.

If the PC to be used as a server is running Windows **Workstation**, the listener must be manually installed by executing the command: lcolist -I. For further information refer to Set Up Listener on the Data/Application Server in the *LANSA for the Web Administration Guide*.

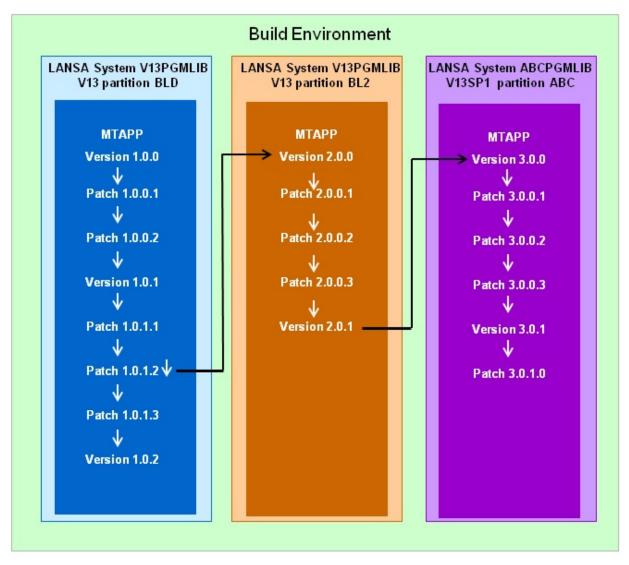
You must check the status of the listener before a PC can be used as a server.

3.2 Lifecycle of an Application

Application deployment isn't over after the application has been successfully installed on the target system. In fact this is just the beginning.

Your applications may receive minor or major updates, new versions of LANSA may be installed or new versions of the application may be required to be installed alongside existing applications. These and other issues determine how your software needs to be managed and evolved though the life time of the application.

The following scenario details one method for managing the development of your software:



In this scenario three different development environments are required to

support development of the application at different versions.

Development of the application commences with Version 1.0.0 on a Version 13 LANSA system V13PGMLIB in partition BLD. After releasing and distributing Version 1.0.0, minor changes to the application are released at different dates as patches 1.0.0.1 and 1.0.0.2. These patches are to be applied to the existing software product. It may be required for users to install both patches, or the patches maybe installable independent of each other.

When more significant enhancements to the application are planned and completed these are released as Version 1.0.1. Following on from here any minor changes to Version 1.0.1 are released as patches starting with Patch 1.0.1.1

At this point, a major set of enhancements are planned for the application. These will be released as Version 2.0.0. It is envisaged that it may not be necessary / advantageous / affordable for all users of the application to upgrade to this Version 2.0.0 immediately so this will be released as a standalone application with on-going support and modifications provided for Version 1.0.1.

To facilitate this all the application objects are exported from partition BLD to partition BL2 on the same LANSA System. Development and released of Version 1.0.1 of the product will continue in partition BLD while the branch of the application in partition BL2 can be modified without impacting Version 1.0.1.

This shared LANSA system with separate partitions means:

- The Application definition is shared between the two partitions. Care must be taken to build the Version and Patches while logged into the correct partition.
- Any files used by the application, assuming they are recompiled, will have a
 different library embedded in the OAM based on the different partition
 Default File Library of the partition where the OAM is generated. This will
 require consideration when installing and upgrading the application as the
 application source code will need to incorporate appropriate use of the
 DEFINE_OVERRIDE_FILE built-in function to direct the file to the
 appropriate library. For more information, refer to
 DEFINE_OVERRIDE_FILE in the Technical Reference Guide.
- Both versions of the Application will be based on the same Version of LANSA. This includes EPCs.

So, this raises the issue, what do you do if you need to continue support for Version 1.0.1 and Version 2.0.0 but there is a new release of LANSA available

and you want to be able to take advantage of the features in the software to enhance your application.

Now you will need to install a second LANSA system ABCPGMLIB at release level V13SP1. Export all the application objects from the most up-to-date version of your application, in this case partition BL2, and import into the new partition ABC on LANSA system ABCPGMLIB. To finalize the setup of the new LANSA system you will also need to copy the Application definition from X_APPS into the new location and optionally the associated Version and Patch definition (for historic reference only). This can be done using the Backup or Restore Applications to a shared location. Copying the Application is important as the GUIDs from the original Application are required if you want to be able to upgrade Version 2.0.0 to Version 3.0.0.

3.2.1 Mapping a File's Library to a Fixed Deployed Schema

☐ Important — When developing an application which includes LANSA files it is important to plan a strategy for managing files updates BEFORE you install the first version of the application.

By default LANSA files are compiled into the partition's Default File Library. When the files are deployed they will, as the default, be created in the database with a schema name the same as the deploying partition's Default File Library. Consequently, if the development environment is moved to another partition with a different Default File Library between Version releases the deployed files will end up in two different schemas (if you use the default processing).

Using the scenario 3.2 Lifecycle of an Application:

Version 1.0.0 the LANSA files compiled in partition BLD will have the partition Default File Library, say BLDDTALIB, embedded in the generated OAM.

Version 2.0.0 is released from a different partition BL2 in the same LANSA system. When the files are rebuilt in this partition the partition Default File Library, say BL2DTALIB, will now be the library embedded in the OAM.

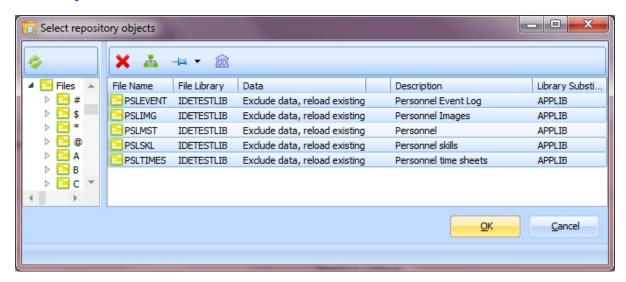
Version 3.0.0 is released from a new LANSA system and yet another partition ABC. When the files are rebuilt in this partition the partition Default File Library, say ABCDTALIB, will now be the library embedded in the OAM.

How should an application be deployed to a consistent schema?

- 1. First, decide what the schema you want to use in the deployed application. This does not have to correspond to a library name in the build environment e.g. LANSALIB
- 2. Create the first version of the application and include the required files. Create a Library Substitution corresponding to the library name selected to be used for the application e.g. LANSALIB.



3. Select this substitution to be used for the files. Refer to Configure the File Library Substitution for more information.



4. Modify the application code to apply the appropriate overrides using the DEFINE_OVERRIDE_FILE Built In Function. This is required to direct the deployed OAM, which has embedded in it the LANSA Library of when the OAM was generated, to the appropriate runtime library (schema). Refer to DEFINE_OVERRIDE_FILE in the *Technical Reference Guide* for more information.

3.3 Version or Patch Upgrade?

When upgrading an Application you will need to consider whether you should do this as a Version (MSI file) or a Patch (MSP file) upgrade. Before making this decision it is important to understand how these respective files will be processed by Windows Installer.

In essence, Windows Installer will process a Version upgrade as a full installation rather than a patch. If an earlier version of the Application is located on the target system Windows Installer will uninstall the older version before installing the new version. If an earlier version of the Application cannot be located on the target system Windows Installer will proceed as if this is a first time install of the Application.

In contrast, Windows Installer will process a Patch upgrade by first determining the version of the application is eligible to receive the patch. If the application is eligible to be upgraded the changes contained in the patch are applied over the top of the existing application. The main advantage of a patch install over a version install is that a patch only contains changes to your application and therefore is usually much smaller.

Patches are typically designed to be uninstallable. Database changes may be complex to manage and thus are usually delivered by a Version upgrade. Database changes are particularly complex when uninstalling. Firstly, only one user of a database must uninstall the database changes. So, database changes are NOT backed-out automatically, you must specifically request it. Secondly, the standard uninstall processing provided by Windows Programs and Features does not provide a user interface, so you cannot request that database changes be backed out. You must use the command line instead.

By default a patch will not uninstall database changes. To do so or not to do so is a complex subject. If a Patch contains database changes you must fully inform your administrators how to back out the patch, or disable patch uninstall in the Deployment Tool package.

In order to uninstall database changes, misexec must be executed from the command line:

For example:

msiexec/package

C:\DEV\TRUNK\WORK\X_WIN95\X_LANSA\X_APPS\PATCHDB\PATC| us.msi /uninstall

C:\DEV\TRUNK\WORK\X_WIN95\X_LANSA\X_APPS\PATCHDB\PATC

us.msp SUDB=1/qb

The parameters **SUDB=1**/**qb** are optional. If they are not specified a dialog will be displayed and you can manually check **Setup DB** on.

Note, these are the functional steps. You must fully consider your specific situation before rolling back database changes. If a database is shared, only one of the users should rollback the database changes. All the rest perform the default uninstall of the patch from Programs & Features.

Alternatively, to back-out a Patch which contains database changes, another Patch would need to be created to restore the database objects back to their original state.

Reasons to consider using a Version upgrade:

- Name of the Application has been changed
- Requirements to allow users to have parallel installations of the Application, with earlier versions and latest version on the same system
- Major change to functionality in Application since last version release
- Major upgrade of LANSA applied since last version release
- Need to install latest version as fully functioning Application
- Modifications to software include database changes.

Reasons to consider using a Patch upgrade:

- Provide enhancements to existing functionality
- Provide new functionality (excluding database changes)
- Changing, removing or adding shortcuts
- Agile development requiring frequent deployment of a small set of changes
- Apply bug fixes to rectify problems with your application.

Note: Patch install and uninstall use the logic in the Version install. Thus, new features for a Patch added in a LANSA release require a new Package Version to be deployed first before you can use the new LANSA features to patch your application.

This also means that patching Versions created with a previous version of LANSA may not work correctly as the Patch code may not be compatible with the Version code. It is therefore highly recommended to NOT upgrade the LANSA configuration which built the Version. What you need to do is to retain a separate Build Configuration for each Version of LANSA which you need to support in the field.

If your application will retain users on both LANSA Version 13.0 and 13.1 for example, then you will need to retain a Configuration for both 13.0 and 13.1. This is good practice anyway, to ensure that as little as possible changes between different releases of your application. Of course you can insist that all users upgrade to the latest Version of your application if they want to receive updates. Then you need only maintain the one Build Configuration.

3.4 Per-User or Per-Machine Install?

Windows Installer can install an application, or an update, for all users of a computer (per-machine) or for a particular user (per-user) based on the access privileges of the user and the operating system. When defining the deployment package the context of the update install needs to be considered. If the contexts of the application and update are different, the application may not be updated as expected.

3.5 Digital Signatures

To utilize digital signatures, an appropriate entry must be entered in the Company details associated with the Application. These details are entered in the Signing Tool Command Line. The format and syntax of the command will be determined by the signing tool utilized.

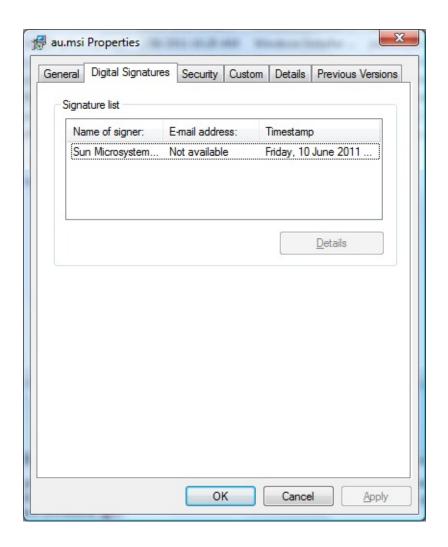
The specified command is executed during the build processing to ensure the resulting MSI or MSP file is digitally signed.

3.5.1 Why digitally sign MSI and MSP files?

If the generated MSI or MSP file is signed by a validated digital certificate the following warning will not be issued when downloading from the internet:



Digitally signed files also detail the party responsible for the production of the file in the file properties.



3.6 Installation Mode

The MSI or MSP files produced by the Deployment Tool build process are already associated with the Microsoft Windows Installer (msiexec.exe) which manages the installation defined by these files. As the files are associated with the Microsoft Installer, the file installation can be started by simply double clicking the files in Explorer.

Alternatively the MSI or MSP files can be installed via the command-line by supplying the MSI or MSP file name as a parameter to the msiexec.exe command. Various parameters are available to control how the install is processed including unattended install, quiet install and verbose install modes.

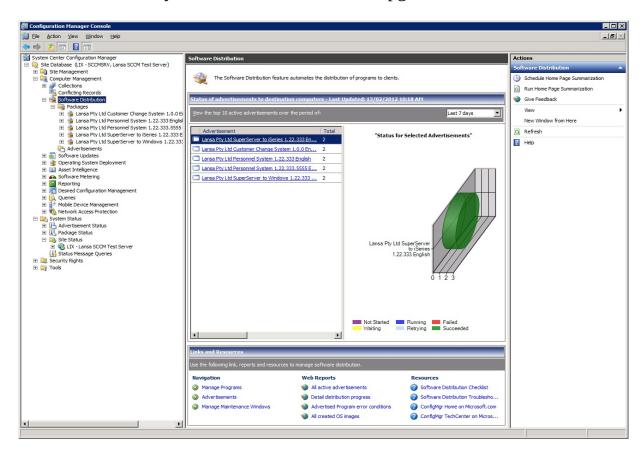
In addition there are switches associated with the msiexec.exe command to control the logging mode.

Note: Regardless of the logging mode used on the msiexec.exe command, all Deployment Tool installations will automatically log their installation behavior (for example in MSI2ce4c.log) to a randomly named file in the end user's defined %TEMP% folder.



3.7 SCCM Integration

The MSI and MSP files produced by the Deployment Tool may be integrated into Microsoft Systems Centre Configuration Management or Systems Management Server tools. These products provide a sophisticated facility to distribute application software within a corporate environment and also provide an alternative facility to LANSA's Just In Time upgrade.



4. Getting Started with the Deployment Tool

Deployment Tool Quick Tour

The Deployment Tool has two main windows and from these windows you will access all the options, settings and commands used with the deployment tool.

After you start the Deployment Tool (see below) you will see the 4.1 Deployment Tool Main Window.

When you select a *Version*, *Patch* or *Template* to work with, you will see the 4.2 Package Maintenance Window.

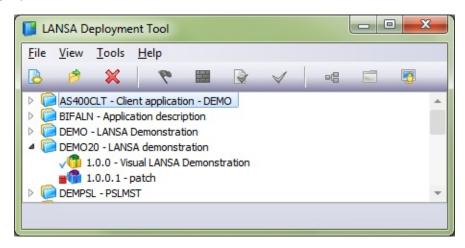
• Context 4.3 Help is available from within the Deployment Tool.

Start the Deployment Tool

Start the Deployment Tool by selecting the *Tool* ribbon in Visual LANSA then select the Deployment Tool .

4.1 Deployment Tool Main Window

When the Deployment Tool is opened, the main window is launched. A tree format displays any existing Deployment Tool Applications and the Packages within them.



From this window you will create, delete, organize and access your Applications for maintenance.

4.1.1 File menu 4.1.2 View menu 4.1.3 Tools menu

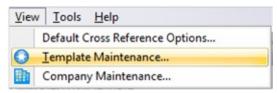
4.1.4 Main window toolbar 4.3 Help

4.1.1 File menu



Use the File Menu to Exit the Deployment Tool. Alternately use Alt+X or close the Main Window.

4.1.2 View menu



Options on the View Items are not specific to an Application.

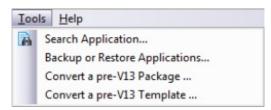
Default CrossReference
Options
Set the default values, by object type, to be used as cross reference settings in a package.
These settings apply across all applications.
Refer to 5.2 Default Cross Reference Settings.

TemplateMaintenance

Create, edit or delete Template definitions. An alternative way to create a deployment template is to save a package definition as a template.

CompanyMaintenance
Create or edit Company details. Alternatively Company details can be access for maintenance when creating an Application.

4.1.3 Tools menu



Items on the Options menu are not specific to an Application or Package.

Search **Application**

Search one or all applications to check which packages

included an object.

Refer to Search Application.

Backup or Restore

It is recommended to backup any Applications which have been released. This will provide a controlled version of the **Applications** released Application which can be restored in the event of any

corruption of the Application. Refer to Backup or Restore Applications.

Convert pre-V13 **Package** The package structure has been modified in LANSA V13. To re-use any predefined packages they must first be converted to a compatible format. Any predefined packages must include a

.ldl file definition to be eligible to convert.

Refer to Convert pre-V13 Package

Convert pre-V13 **Template** The template structure has been modified in LANSA V13. To re-use any predefined templates they must first be converted to a compatible format.

Refer to Convert pre-V13 Template

4.1.4 Main window toolbar



The options available on the main window toolbar are available on the package, and in some cases application context menus (from a mouse right click).

New Create an application.

The package option is only available if an entry is

selected in the tree view.

Refer to Create an Application.

Open an application, Version or Patch. Refer to Open a

Package

The open option is only available if a package is selected

in the tree view.

Delete Delete application or package. Refer to Delete an

Application? or Delete a Package.

Check Perform a pre-build check of a package to verify

everything required for the package settings/options has

been supplied.

Build When the package definition has been completed, use the

build option to generate the package.

■ View Log
Review the last build log for a selected package.

✓ Distribute After a package has been successfully built it is available

to be distributed. When a package is flagged this indicates the package is complete and ready for

distribution.

Review Objects View the objects included in a package definition.

Objects directly selected for inclusion in the package can be viewed, or a separate tab details the complete list of objects including cross referenced objects. The list of

objects can be saved to a file.

Go to folder Opens the folder containing the Application definition

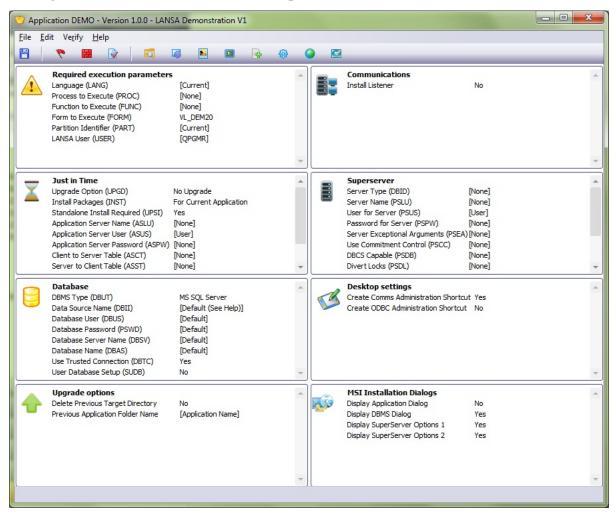
and related files in Windows Explorer.

Communications Include or change a Host Route table for an Application. **Administrator** When you first select the Communications Administrator

When you first select the Communications Administrator option a copy of your current LANSA system's Host Route (lroute.dat) and Listener (listen.dat) files will be added to the Application definition. These versions of the Host Route and Listener files can be modified as required.

4.2 Package Maintenance Window

When you select a package (by double clicking on it in the Application tree) the *Package Maintenance Window* will open.



From this window you access all the options and settings used to define a Version or Patch. From this window, you will also Save your options and settings, Build your package and can optionally save it as a Template (if you wish to use the package settings as the basis of another package).

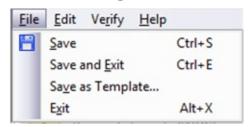
If you use a Deployment Template as the basis of a package, the creation of a package may require you to change very few options. Options shown in gray indicate that the option has been pre-set by the Deployment Template on which the package is based.

The options and settings are accessed from either the groups of scrollable options in the body of the window (these subsets are called the *Installation*

Options) or the toolbar buttons. When you select one of the subsets or toolbar buttons, the relevant dialog window is opened for you to make your selection. Each of these options and settings is described in this guide.

- 4.2.1 Package File Menu 4.2.2 Package Edit Menu
- 4.2.3 Package Verify Menu 4.2.4 Package Maintenance toolbar

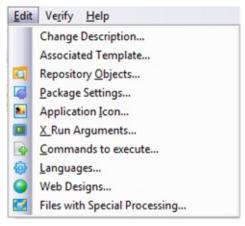
4.2.1 Package File Menu



Use the File Menu to save and/or exit the current package and return to the main window.

Save	Save the package definition. (Ctrl + S)
Save and Exit	Save the package definition and close the <i>Package Maintenance Window</i> . (Ctrl + E)
Save as Template	Save the current package as a deployment template
Exit	Close the <i>Package Maintenance Window</i> . (Alt + X)

4.2.2 Package Edit Menu



The Package Edit Menu in the *Package Maintenance Window* provides access to options for configuring and defining the current package. Most of the Package Edit Menu options are also available on the *Package Maintenance toolbar*.

Change Modify the package description for easier identification **Description**

Associated Template

This option is only shown if the current package has an associated template.

Verify and if necessary remove the template associated with the current package.

Refer to Associated Template.

Repository
Objects

Select LANSA and non-LANSA objects to be included in the package.

Package Settings

The *Package Settings* determine the additional processing that will occur as part of the package build processing. Each option selected will include or omit specific features with the package during the build.

- **Application** Select an icon to identify the Application when a shortcut is **Icon** created on the target machine during package installation.
- **X_RUN** When a package is built any X_RUN arguments defined will be incorporated into a start file.

The most commonly used arguments are covered by other options in the *Package Maintenance Window*, however,

additional X_RUN information can be provided to change the behavior of your application at run time.

Commands Enter any operating system level commands you require to run **to execute** before or after the package is installed on a target machine.

Languages The *Package Languages* dialog allows you to select the installation language(s) to be used when a package is installed.

Web Indicate the Languages and Technology Services to be included when the package includes WAMs.

Files with Special Files with the Package. These files are installed and special processing may be executed during the processing dependent on the file type and instructions.

4.2.3 Package Verify Menu



Typically you would only use entries on the Package Options menu (aside from *Change Package Description*, *Template associated with Package* and *Communications Administrator*) when the Package definition is complete.

- **Check** Verify the package as a pre-build operation. (This is quicker than attempting a build only to have it fail).
- **Build** After a package has been saved you can build the package based on the current definition.
- **View** Review the log file associated with the last build of the current **Log...** package.

4.2.4 Package Maintenance toolbar

All the options available on the *Package Maintenance toolbar* are also available in the 4.2.2 Package Edit Menu and the 4.2.3 Package Verify Menu.



The *Package Maintenance Window* contains a toolbar for easy access to the actions on the Menus. If a toolbar icon is shown as inactive (that is, grayed out) it may be because it is defined as "hide" in the deployment template associated with the current package. Refer to Deployment Template Options & Settings to learn the control a template has over package options and settings.

The tools on the toolbar are:

Save Package	Save your Package selections. For information refer to
	Save a Package.

Check package	Verify the package as a pre-build operation. (This is
	quicker than attempting a build only to have it fail).

Build Package	After a package has been saved you can build the package
	based on the current definition.

View log	Review the log file associated with the last build of the
	current package.

👩 Select	Select Objects, such as Components, Functions and
Repository	Process to be included in the package. For information
Objects	refer to Select Repository Objects.

Package	Select the components to be included with the package, for
Settings	example, <i>Deploy with Component Support</i> . For
	information refer to Settings.

Application Icon	Allows you to specify the icon to be used for the
icon	Application's desktop shortcut. For information refer to Application Icon.

X_RUN	X_RUN parameters are additional arguments for the
arguments	X_{RUN} command. For information refer to X_{RUN}
	Arguments.

Commands to Select Command to specify OS level commands to be

execute before and after import	executed before and after installation of the package. For information refer to Commands to Execute.
• Languages	Select <i>Language</i> if you are using multiple languages. For information refer to Languages.
Web Designs	Indicate the Languages and Technology Services to be included when the package includes WAMs.
Files with Special Processing	Include Special Files with the Package. These files are installed and special processing may be executed during the processing dependent on the file type and instructions.

4.3 Help

You can use the function key **F1 to open context sensitive help fr**om anywhere in the Deployment Tool. **F1** will go directly to the relevant topic in the *LANSA Application Deployment Tool Guide* (this guide).

Alternatively, you can open and browse through the whole *LANSA Application Deployment Tool Guide* from the *Help* and then *Contents* menu.



5. Configure the Deployment Tool

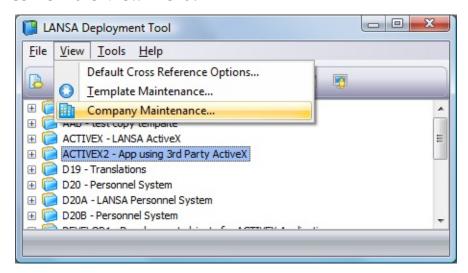
The Deployment Tool is available to use as soon as your LANSA system is installed.

After installation you may want to perform the following configuration steps to enhance your Deployment Tool experience.

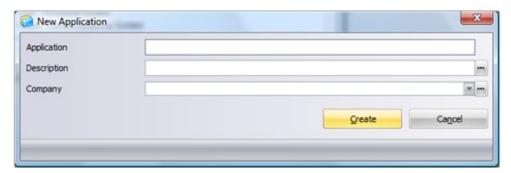
- 5.1 Set up Company Details (required)
- 5.2 Default Cross Reference Settings (optional)

5.1 Set up Company Details

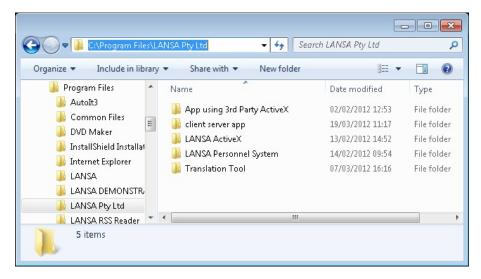
Company information is required when creating an Application. Company details can be set up or modified at any time by accessing the Company Maintenance from the View Menu.



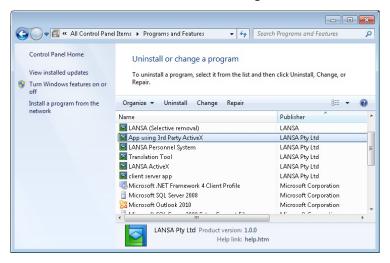
Company details can also be set up or modified using the prompt button on the Company when creating an Application.



Company information is used at installation time to group applications in a common Windows program folder. Multiple Companies can be created if required and can be selected when creating the Application.



The Company name is also used to indicate the publisher of the Application.



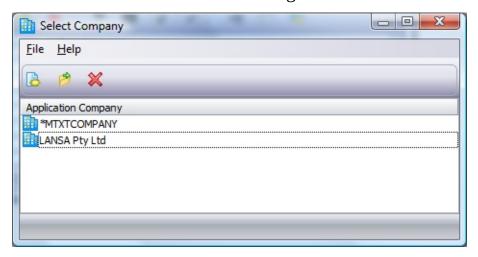
Also see

- 5.1.1 Company Maintenance
- 5.1.2 Create a Company

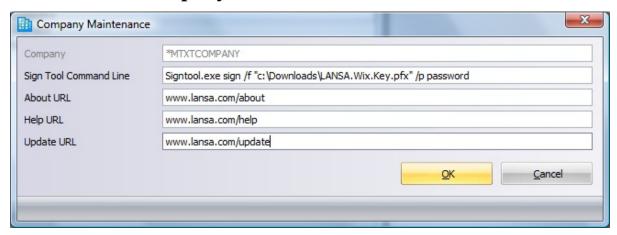
5.1.1 Company Maintenance

To review, change or delete a company open the Company Maintenance dialog and select the company to work and the appropriate action from the toolbar. An alternate way to open a company is to double-clicking the company in the list of companies.

Companies can also be created from this dialog.



5.1.2 Create a Company



Company Required.

The Company Name must be entered. It can be entered as text or a Multilingual Variable can be nominated if the Application will be deployed in multiple languages with a variation on the company name which is language dependent.

Signing

Optional.

Tool Command Line

Any command entered in this value will be used to automatically digitally sign the MSI and MSP files generated when a Version or Patch is built. The Signing Tool Command Line allows the specification of the parameters to be passed to signing tool, for example the location of a Digital Certificate. The Signing Tool Command Line must have the format:

signtool [command] [options] [file_name | ...]

About Optional.

URL

If entered the URL will be displayed in the Windows Package Maintenance Window when the Application is installed on an end user's computer.

Help Optional.

URL

If entered the URL will be displayed in the Windows Package Maintenance Window when the Application is installed on an end user's computer.

Update Optional.

URL If entered the URL will be displayed in the Windows Package Maintenance Window when the Application is installed on an end user's computer.

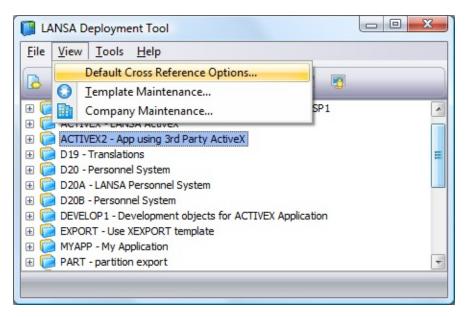
5.2 Default Cross Reference Settings

Typically, each Deployment Package you create will include a selection of LANSA and/or non-LANSA objects. Cross referencing allows you to select an object (for example, a Function or Component) for inclusion in a Version or Patch, and then have all, or selected types, of the related executable and definition objects automatically included at build time.

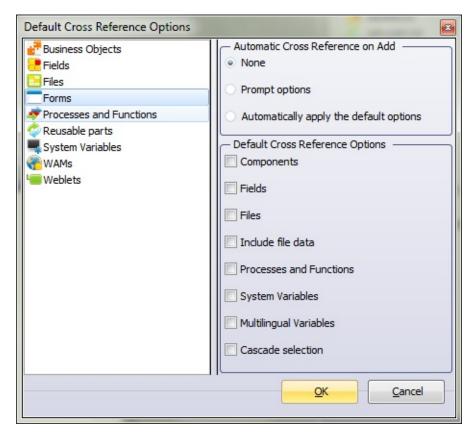
Use the Default Cross Reference Settings to define the initial cross reference behavior and options to be used in each package. Any default options defined here can be overridden when editing the Version or Patch, but the investment of time to set up your default values correctly can by-pass several repetitive steps in the definition process.

Default Cross Reference Options will apply across all Applications defined in the Deployment Tool.

Default Cross reference behavior is defined in the Default Cross Reference Options dialog opened from the View menu on the Deployment Tool's main window.



Each object type included in the left hand side of the Default Cross Reference Options has their own default settings. Select each object type to review the default settings.



The options selected in the *Automatic Cross Reference on Add* section of the dialog control how the default cross reference options are displayed and applied within an individual package. These are:

None

Cross References will only be assigned for the objects included in a package if you manually select the cross reference icon 1 and options within the Package's Repository Object Selection dialog.

Prompt options

When you add objects to a package using the Repository Object Selection dialog and press OK to confirm your object selection you will be presented with Cross Reference options for each object type you have included in the package. The initial settings are defaulted based on the *Default Cross* Reference Options but can be modified to suit the individual package requirements.

apply the default options

Automatically The default values for the object type, as set up in the *Default* Cross Reference Options section of the Default Cross Reference Options dialog, will be automatically applied to any objects included in a package. The default values can still be

overridden in a package by selecting the cross reference icon in the package's Repository Object Selection dialog and modifying the settings as required.

The settings in the *Default Cross Reference Options* section of the dialog indicate the default settings for an object type to be applied to a Version or Patch. These settings can be overridden when editing a specific Version or Patch.

It is recommended to not use the option Include File Data as this will then be the default option used not only in the first Version but also in any subsequent Versions and Patches and as such may produce undesirable results.

Refer to Cross Reference Settings for more information.

6. Applications

The first step in deploying your LANSA-developed application is to create an application, or applications. How to do this is described in 6.1 Create an Application.

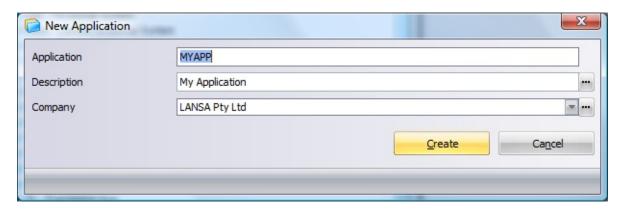
If you are new to the Deployment Tool, refer to &a href lansa022.chm::/Lansa/l4wdplb2_0060.htm&Plan your Deployment &a& before you get started.

The Application is also important in determining whether the Host Route Table and Listener Table are deployed with the Application. Refer to 6.4 Include Communications Tables for further information.

You can also 6.2 Change an Application's Details and 6.3 Delete an Application.

6.1 Create an Application

In the main window, select the *New Application* tool on the toolbar. *Alternatively you can select New* in the tree menu.



Application Required.

Enter a name for the Application in eight characters or less. The first character must be an alphanumeric character and the name cannot contain embedded blanks. Applications cannot start with the letter X.

Application Required.

Enter a description that you can easily find and understand when you have a number of Applications. It is also the name that the user will see when the application is installed. It can be entered as text or a multilingual variable. If the Application will be deployed in multiple languages, use a multilingual variable so that the language is appropriate for the end user.

This Application description will be used when installing and uninstalling the Application. Thus it is essential that a different description is used for each language if more than one language needs to be installed on a single PC. This is because using the same language description will cause some parts of the installation to overwrite each other which will cause uninstall and upgrade issues.

Company Required.

Select a Company to be associated with this Application. To define a Company use the ellipse button to access the Company

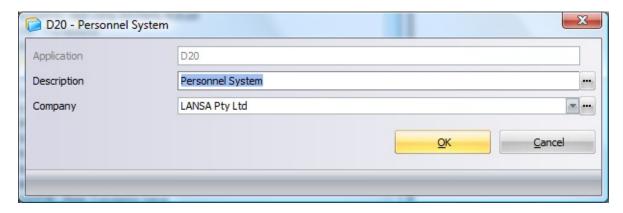
Maintenance. The Company and associated details are used during the build processing as well as during installation and uninstallation of the Application.

Refer to Set up Company Details for more information.

Press *Create* once you have entered the information to define the Application. You will be automatically prompted to Create a Version within the new Application.

6.2 Change an Application's Details

The Application Name cannot be modified but the associated description and company are editable.



Application Not maintainable.

The Application Name cannot be changed.

Description Required.

Modify the Application description as required.

It is not recommended to change the Application Description after the Application has been distributed as the description is used during the installation and to identify the installed Application.

Company Required.

The Company Name must be entered. It can be entered as text or a Multilingual Variable can be nominated if the Application will be deployed in multiple languages with a variation on the company name which is language dependent.

It is not recommended to change the Company after the Application has been distributed as the Company is used during the installation and to identify the installed Application.

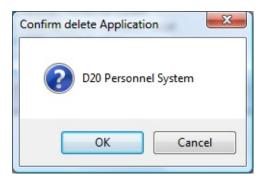
Refer to Set up Company Details for more information.

Press *OK* to save any changes to the Application or *Cancel* to close the dialog without saving changes.

6.3 Delete an Application

Select the Application to be deleted and press the *Delete* tool in the toolbar or choose the Delete command from the *right-click* menu.

You will be prompted to confirm the deletion.



When you delete an Application, all Versions and Patches within the Application are deleted. Refer to Backup and Restore Application Definitions.

6.4 Include Communications Tables

To include or change a Host Route file for an Application, press the Communications Administrator tool in the toolbar.

Initially, the host route (lroute.dat) information is copied (lroute.dat) from the current LANSA system. This copy can be modified as required for the specific Application.

Listener information can also be deployed by selecting to view the listener details in the host route and modifying as required.

Refer to Deploy LANSA Communications for more information.

7. Versions and Patches

Before you select the objects and specify the settings and options for a Package, you must first create one by specifying the name and description (as a minimum).

From the Package menu on the Deployment Tool's main window, you can:

- 7.1 Create a Version
- 7.2 Create a Patch
- 7.3 Open a Package
- 7.4 Delete a Package
- 7.7 Check a Version or Patch
- 7.8 Build a Version or Patch

Check the 7.8.1 Package Build Log

7.9 Package Distributed

Additional options are available when the Package is open for editing or review, including:

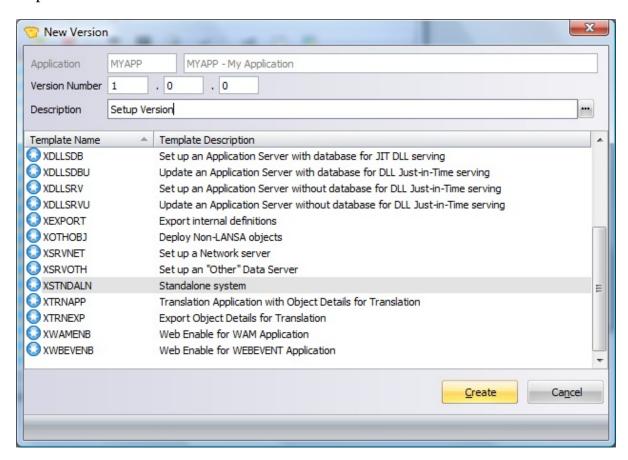
7.6 Save a Package

Save a package as a template as shown in Create a Deployment Template. and

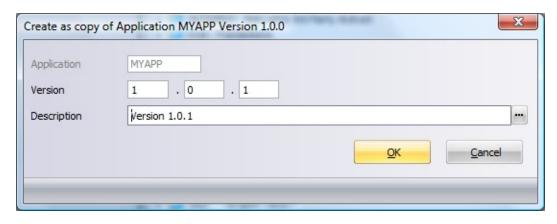
Other 8. Options & Settings, to allow for the configuration of the Package.

7.1 Create a Version

The initial Version in an Application is created when the Application is created. After confirming the Application details the New Version dialog is presented.



Subsequent Versions in the Application are created by copying the original Version. The *Create Next Version* option is available in the context (right click) menu when a Version is selected in the Main Window tree. The Version will be created under this selected Application.



Application The Application containing this Package. This value is prefilled and cannot be modified.

Version

Enter the next Version number. Version number must increase for subsequent releases. A recommended approach for naming is .

<major version number>.<minor version number>.

<bul><build number>

Description Enter a description.

Template Name

A template can only be selected when creating the first Version in an Application.

Select a template to use as the basis of the Version, or select the "no template required" entry in the list.

Deployment templates will save you time by providing some preset options and hiding unnecessary options.

To save time, some templates have been supplied with the Deployment Tool. For a list of the templates, refer to Supplied Templates.

Press *OK* when you have completed these entries. The *Package Maintenance Window* will be opened so that you can select the options, settings and objects to be included in your new Version. Go to 7.3 Open a Package.

7.2 Create a Patch

A Patch is created by copying a Version or Patch in the same Application. This copied Version or Patch forms the basis of the Patch definition.

The *Create Patch* option is available in the context (right click) menu when a Version or Patch is selected in the Main Window tree. The Patch will be created under this selected Application.



Application

The Application containing this Package. This value is prefilled and cannot be modified.

Patch

Enter the next Patch number. Patch number must increase for subsequent releases. The Patch number has a direct relationship to the Version or Patch it was based on, for example Patch 1.0.0.1 would be the first patch against Version 1.0.0. A recommended approach for naming is:

<major version number>.<minor version number>.

<build number>.<patch number>

Description

Enter a description.

Uninstallable Allows a patch to be uninstalled. The uninstall process will roll back the LANSA component DLLs changed by the patch. If the Patch install updates the database, any changes to the database will not be uninstalled.

Press OK when you have entered Patch details. The Package Maintenance Window will be opened so that you can select modify Files with Special Settings and the objects to be included in the Patch. All other settings and options

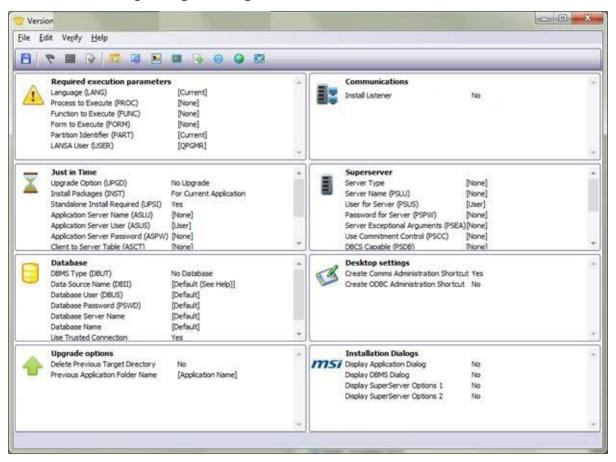
cannot be modified and reflect values of the original copied Version or Patch.

7.3 Open a Package

In the main window, select the Version or Patch to maintain then use the *Open* tool on the toolbar, or from the context (right click) menu, to open the selected Version or Patch. A package can also be opened to edit by double clicking the package in the Main Window tree.

When you Create a Version or Create a Patch it is automatically opened to edit.

The *Package Maintenance Window* is the starting point for selecting all the options, settings and objects in your Package. These options and settings govern the behavior of a package during its installation and execution.



The options on this window have been grouped so that, for the majority of groups, a whole group can be ignored and default values assumed if they are not relevant to your package. For example, if the package is not using the *Just in Time* facilities, the defaults can be accepted, and you can then ignore the whole *Just In Time* group. Similarly, if you are not deploying to a local database, the

Database options can be ignored.

To modify the base definition of a package, refer to:

7.3.1 Edit Installation 7.3.3 Change 7.3.4 Associated Options Description Template

For details of each group, go to:

Required Execution Parameters

Just in Time

Database

Upgrade Options

Communications

LANSA SuperServer

Desktop settings

Installation Dialogs

7.3.1 Edit Installation Options

All of the values displayed in the *Package Maintenance Window* are available for review, and in most cases can be changed. If the package was created

- using a deployment template, the template can control whether an option is display only or can be changed.
- not using a deployment template, default values will be displayed.

Once you change an option, it will display with a gray background and black text to indicate that a change has been made.

Options pre-set to a specific value that cannot be changed, as pre-determined by a template, are displayed with a gray background with gray text.

Options that must be entered, as pre-determined by a template, are displayed with a gray background with black text. Refer to Deployment Template Options & Settings for more information.

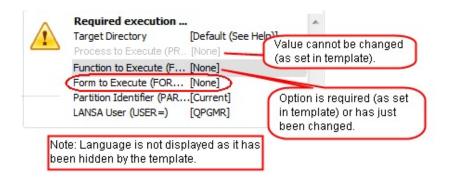
As many of the options equate directly to specific execution parameters, the equivalent X_RUN argument is included at the end of the option's description. For a full explanation of the use of the X_RUN command and parameters you should review Using the X_RUN command in the *Technical Reference Guide*.

Important: The default values for installation options are enclosed in square brackets; for example, Language (LANG) has a default value of [Current]. **DO NOT enclose your installation option values in square brackets**. If you do, the value will be ignored and will default back to the Deployment Tool's default value for that option when the package is installed.

To set a value for an option on the *Package Maintenance Window*:

Step How to do it

- 1. Create or Open a Version. Notice the font and highlighting associated Double with the displayed Package Maintenance Window options. There will click only be variation from the standard black text with white background on an if the package was based on a template.
- option Double click on an option to review the value and appearance settings. For example, double-click the Form to Execute option:

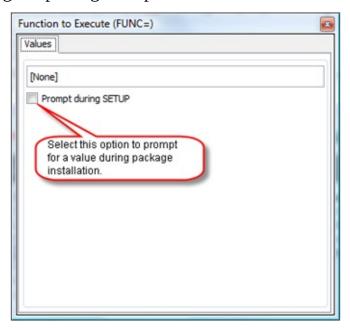


Refer to 8.1 Installation Options for information on specific options.

2. Set A dialog box will be displayed for you to select or enter your required up the values and options.

Values The *Values* tab allows you to specify a default value to be applied default when the package is installed.

If the option is a free-form entry, like Form to execute (FORM), an edit box will be displayed. Change the value to the value you want applied during the package setup and installation.



In this case you can also decide if you want this value to be prompted, thus enabling it to be changed during installation. Some options also include Prompt during X_START if this is relevant.

DO NOT enclose your installation option values in square brackets.

Prompt during Installation

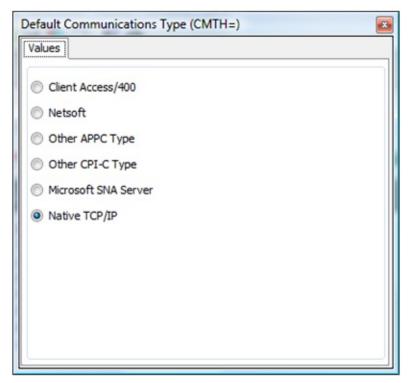
When installing the package you will be prompted to enter or confirm the value for this option. If the value is a password (i.e. the PSWD, PSPW, and ASPW parameters) then the default value is set to '*password_optional'. If there is no value for the parameter at the time of the installation, then the default value is set to '*optional_data_'. For more details, refer to the 8.1.8 MSI Installation Dialogs and Commands and Special Variables in the *Technical Reference*.

Prompt during X_START

When running the application using an X_RUN (typically from a desktop icon) you will be prompted for X_START values when the application is launched. The values entered will be used as default values on the X_START prompt. In normal circumstances, this option should not be selected.

Note: If you nominate to *Prompt during Installation* for an option you must also set the corresponding dialog Installation Dialogs display to Yes, otherwise the option will NOT be prompted.

When the option has a predefined list of values, a series of radio button is displayed in the values tab.



Tip: You can open consecutive option dialogs without closing the previous dialog. The current dialog will be replaced with the newly

opened dialog.

7.3.2 Edit Settings

The package's settings are accessed from the *Package Maintenance Window*'s toolbar, and consist of:

- 8.4 Application Icon
- 8.5 X_RUN Arguments
- 8.6 Commands to execute
- 8.7 Languages
- 8.8 Web Designs
- 8.9 Files with Special Processing

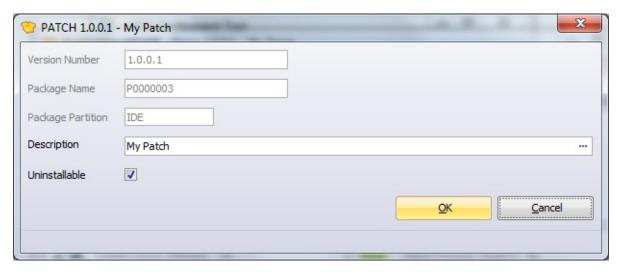
Settings can be selected in any sequence.

All the package's settings are available for review, and in most cases, can be changed. If the package was created using a deployment template, the template can control whether a setting is only for display or if it can be changed. Where a deployment template has not been used, default values will be displayed for a new package.

Options pre-set to a specific value and not changeable, as pre-determined by a template, are displayed with the message *Read only mode - Changes to the values cannot be saved*. To modify the values, the reference to the template must be removed. Refer to 7.3.4 Associated Template.

7.3.3 Change Description

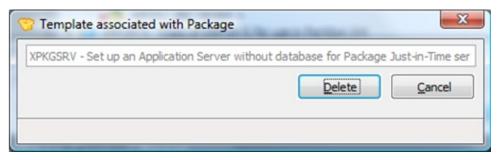
Select *Change Description* from the *Edit* menu on the *Package Maintenance Window* to change the current Version or Patch description. The option to allow a Patch to be Uninstallable can also be modified.



7.3.4 Associated Template

The Associated Template option is only available if there is a template associated with the current Version or Patch. It is opened from the command on the *Edit* menu in the Package Maintenance Window.

Use this option to verify what template the package has been based on, and if necessary remove the template association to allow greater flexibility in the package definition. The associated template can hide or restrict access to some of the Settings and options.



Packages based on a Template

In many cases, you will be able to limit the number of options you need to consider by basing your Package on a deployment template when the package is created. A Template can contain pre-set options and even have some complete subsets of options disabled or hidden.

The *Package Maintenance Window* reflects the options specified for the selected template so the *Package Maintenance Window* you see may not be identical to the one in the following example. Pre-set options are displayed with a gray background and gray text.

Note: If a package uses a deleted template, it will simply ignore the template settings when loaded. This means that the package will now display in its default state, with all options visible and changeable.

For more information refer to Deployment Template Options & Settings.

7.4 Delete a Package

★ Highlight the package to be deleted in the list in the Deployment Tool's main window and press the Delete icon.

The Delete icon will only be available to use if the package is closed. If a package has been released, you will be prompted for confirmation of the deletion.

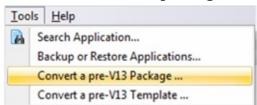


Once the Delete option is used, the package definition and all physical objects will be deleted.

There is no back out facility for a package deletion.

7.5 Convert a Pre-V13 Package

Use this option to convert a package in a pre-V13 LANSA environment to be used as the base Version of your Application in Version 13. In the main window, select the Convert a pre-V13 Package option from the Tools menu to locate and convert a package defined in an earlier version of LANSA.

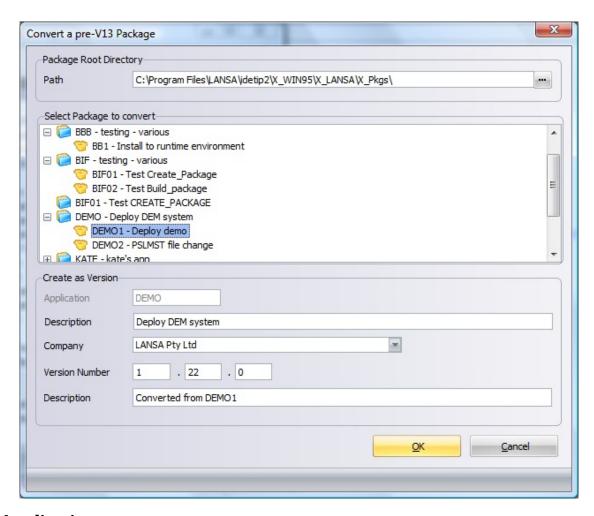


Before converting any package ensure you have understand The Deployment Model and the changes in the deployment paradigm in Version 13.

Important – If you have several packages in pre-V13 LANSA to install a single Application select the most complex package to convert. Any additional objects, options or settings can be added after the conversion is complete.

The conversion interface is a four step process:

- 1. Enter path where package to be converted exists.
- 2. Select a package to convert.
- 3. Confirm and modify details as required.
- 4. When details are complete select OK to convert the selected Package.



Application Application is pre-filled from the copied Application and cannot be modified.

Description Enter an Application description or use the copied Application description. This is important as it is used to identify the installed Application.

Company Select a Company to be used as the distributor of this Application. If no Companies available create Company details using the Company Maintenance on the View Menu. Refer to Set up Company Details for more information.

Version NumberEnter an appropriate Version Number to indicate the software release.

Description Enter a description for the Version. This is for information only as is not used in the installed Application.

7.6 Save a Package

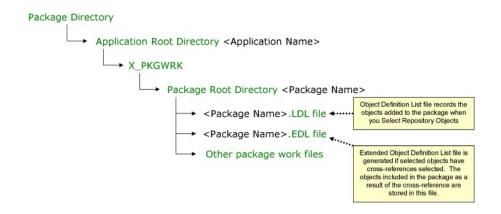
Save a package by selecting the *Save* tool on the *Package Maintenance Window* toolbar or select the *Save* command in the Package menu.

When you save a package, the options, settings and object lists specified are stored in an LDL file which is written to the package work folder.

Once you have saved a package, you can build it as described in 7.8 Build a Version or Patch.

Note: Until a package is saved for the first time, a minimum set of information is saved to the disk. If you do not save the package, the next time you open it, you will only see the defaults (that is, the minimum set of information that was saved).

The saved package details are stored in a working directory structure:



7.7 Check a Version or Patch

From the Deployment Tool main window or the Package Maintenance Window toolbar, select the *Check* option. The Check option is also available on the context (right click) menu in the *Deployment Tool* main window.

This option is only available after you have saved a Version or Patch and before attempting to build. If you make further changes, the Build and Check options are disabled until the package is Saved again.

This is an optional step to perform before attempting to build a package. Use the Check option to identify any issues in the package, allowing them to be rectified before the package build is initiated. This can save considerable time at the package Build stage.

important - Before attempting to build a package, it is recommended that you correct the conditions for all warning messages returned by this package check process.

7.8 Build a Version or Patch

Once a Version or Patch has been Saved, and optionally passed the pre-build Check process successfully, you can build it.

To start the build process, select the *Build* icon in the Deployment Tool main window or *Package Maintenance Window* toolbar. The build option is also available on the context (right click) menu in the Deployment Tool main window.

The Build process uses the definitions from the associated files (created when the package is saved) to create the required folders, run any necessary exports, and copy the required objects into the package structure. If you have included any Cross Referenced objects in the package, they will be resolved during the build.

The final phase in the build process generate WIX XML source files which are then compiled and converted into MSI and MSP files by the WIX toolset. Package build is not supported on a network client installation.

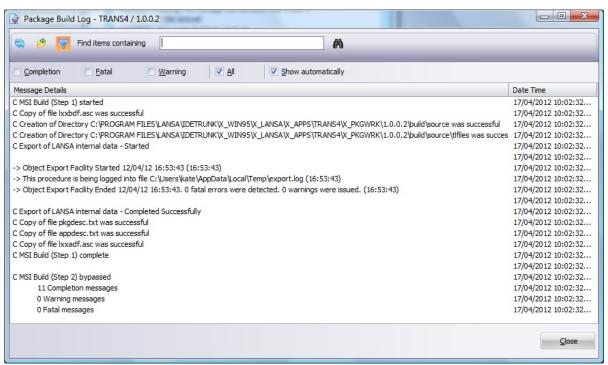
7.8.1 Package Build Log

Whenever a package is built, a build log is created containing all the messages generated during the package build process.

Select *View Log* from the *Deployment Tool Main Window or Package Maintenance Window toolbars*. The *View Log* option is also available on the context (right click) menu in the Deployment Tool main window.

This option is only available when the selected package has been built.

The log displayed will look something like this:



Use the options along the top of the dialog to filter the messages to be displayed. The options selected will be remembered from build to build.

Take special note of the Show automatically option as this will determine whether the View Log is automatically presented immediately after the build has completed.

☆ Important - If a build fails for any reason, look at the Build Log for an explanation.

The actual log file associated with this dialog, lpcreate.log, is stored in the corresponding package directory, $\langle sysdir \rangle \backslash X_Apps \backslash$

<Application>\X_PKGWRK\<Version>\Build\dpcreate.log Details of
any export logs generated during the build process are automatically
incorporated into the lpcreate.log and identified by a prefix of ->.

7.9 Package Distributed

▼ To indicate that a Package has been distributed select *Distributed* from the *Package Maintenance Window* toolbar or context (right click) menu of the selected Package.

A package can only be flagged as distributed after it has been built. Distributing a package means that it is now available for automatic download from a *Just in Time* server. This is also an indication that the package should not be rebuilt without careful consideration of any subsequent Version or Patches in this Application. If the package is rebuilt the new set of GUIDs generated will be incompatible with subsequent Patches.

If your packaging system is the same system that you are using for *Just in Time* deployment, the distributed package can be downloaded immediately. This may not be desirable, so it is recommended that you use different machines for creating and deploying packages.

8. Options & Settings

The *Options* and *Settings* are divided into three main categories. These are:

- The 8.1 Installation Options that are accessed from the main window of the *Package Maintenance Window*.
- The application's objects described in 8.2 Select Repository Objects.
- The 8.3 Settings which are accessed from the *Package Maintenance Window*'s toolbar, and consist of:
 - 8.4 Application Icon
 - 8.5 X_RUN Arguments
 - 8.6 Commands to execute
 - 8.7 Languages
 - 8.8 Web Designs
 - 8.9 Files with Special Processing

Options and settings can be selected in any sequence.

8.1 Installation Options

The installation options, which are specified via the *Package Maintenance Window*, govern the behavior during installation and subsequent execution on the target PC.

For the majority of the installation options, whole groups can be ignored and the defaults used. For example, if the *Just In Time* feature is not going to be used, you can ignore that group of options.

Basing the initial Version of an Application on a Deployment Template will reduce the need to specify requirements even further.

Many of the installation options equate directly to specific LANSA execution parameters, so where applicable, the equivalent X_RUN argument is included at the end of the item's description. You will find a full explanation of X_RUN and these arguments in Using the X_RUN Command in the *Technical Reference*.

The default values for installation options are indicated enclosed in square brackets; for example, Language (LANG) has a default value of [Current]. **DO NOT enclose your installation values in square brackets**. If you do, they will be ignored and will default back to the Deployment Tool's nominated default value for that option.

The Parameter groups required are:

- 8.1.1 Execution Parameters
- 8.1.2 Additional LANSA Features
- 8.1.3 Just in Time
- 8.1.4 SuperServer
- 8.1.5 Database
- 8.1.6 Desktop Settings
- 8.1.7 Upgrade Options
- 8.1.8 MSI Installation Dialogs

8.1.1 Execution Parameters

These are mandatory parameters that must be specified before a Visual LANSA application can be executed. You can specify to launch a Process, Function or Form - not all of them.

Require Indicates if the Application must be installed with Administrator Elevation rights.

Yes – Elevation is required to install application.

No – Elevation is not required to install application.

Default: No

Language Enter the language code to be used when the application is **(LANG)** executed.

> **Default:** [Current] - The language code used when the package is created.

Process to **Execute**

(PROC)

Specify the process that is used to start the application once it is installed. If a Process or Form to Execute is not specified, the Process to be executed will be set to the special process *INSTALL.

Typically Upgrade packages will not need to specify a Process, Function or Form as the information will have been deployed previously.

Unpredictable results will occur if both a Process and a Form to Execute are specified.

This object should be included in the package.

Default: [None]

to

Function Specify the Function that should be called after the application is installed.

Execute (FUNC)

If this option is specified you must also specify a Process to Execute (PROC).

This object should be included in the package.

Default: [None]

Form to Execute

Specify the Form that is used to start the application once installed.

(FORM) Unpredictable results will occur if a Process and Form to Execute

are both specified.

This object should be included in the package.

Default: [None]

Partition Enter the partition to be used when the application is executed.

identifier (PART)

Default: [Current] - The partition used when the package is

created.

LANSA The LANSA user for application execution.

User (USER) Default: [QPGMR]

8.1.2 Additional LANSA Features

Install

Indicates if the listener will be installed.

Listener

Yes – Listener will be installed.

No – Listener will not be installed.

Default: No

Install Web

Indicates if the LANSA for the Web Administrator will be Administrator installed. This is used to configure the LANSA for the Web software on the Web Server and the Data/Application Server.

Yes – LANSA for the Web Administrator will be installed.

No – LANSA for the Web Administrator will not be installed.

Default: No

Install IIS Plugin

Indicates if the IIS Plugin will be installed. The LANSA for the Web IIS Plug-in is an ISAPI based Web serving technology that can be used to deploy LANSA applications using the Microsoft IIS Web Server. The IIS Plug-in is the component used to build the link between the Web Server and the Data/Application Server.

Yes – IIS Plugin will be installed.

No – IIS Plugin will not be installed.

Default: No

Install Web Images

Indicates if the Web Images will be installed. The Web Images are a default set of image files used by LANSA for the Web.

Yes – Web Images will be installed.

No – Web Images will not be installed.

Default: No

Install Java Service Manager

Indicates if the Java Service Manager will be installed. The LANSA Integrator Java Service Manager allows developers to integrate Java programs with their existing LANSA or 3GL applications. It provides a standardized interface for calling Java classes using Java service programs.

Yes – Java Service Manager will be installed.

No – Java Service Manager will not be installed.

Default: No

Install Visual LANSA Framework

Indicates if the Visual LANSA Framework will be installed.

Yes – Visual LANSA Framework will be installed.

No – Visual LANSA Framework will not be installed.

8.1.3 Just in Time

Set the Just in Time (JIT) options if you want upgrades to the application to be automatically deployed from a central server when the end user starts the application. If you are going to distribute changes manually or via DVD or a network, you can bypass these JIT options.

Before you select your Just In Time options and settings, please review the recommendations in Just in Time Upgrade Guidelines.

Upgrade option (UPGD)

This option is used to describe how *Just In Time* upgrades from an Application Server will be performed.

No Upgrade - Do not do any JIT upgrade processing.

New Versions/Patches - Get a copy of any new Version or

Patches.

Default: No Upgrade

Standalone Install Required (UPSI)

This option specifies whether the installation of packages requires all users to be disconnected from the LANSA system. It is added to the start up process for the application when starting it from its Icon.

Yes - The system must have no active users before installing the package. If any user is connected packages will not be installed/upgraded.

No - Users can remain connected when packages are being installed/upgraded. Typically this would only be suitable for Network Client packages, or packages being installed on a Standalone system.

Prompt User - The user will be prompted if any part of the application is active prior to installation/upgrade of packages.

Default: Yes

Application Server Name (ASLU)

Default: [None]

Specify the Name of the Application Server (ASLU) where the

Just In Time updates will be located. Refer to The

PSXX=Parameter name PSLU= in the *Technical Reference*.

Application Server User

Default: [User]

Specify the user name to be used for connecting to the

(ASUS) Application Server. Refer to The PSXX=Parameter name

PSUS= in the *Technical Reference*.

You do not need to specify this option if the value is the

same as the LANSA User.

Application

Default: [None]

Server **Password** (ASPW)

Specify the password to be used for connecting to the

Application Server.

For information, refer to The PSXX=Parameter name PSPW=

in the Technical Reference.

Client to **Server Table** (ASCT)

Default: [None]

For information, refer to The PSXX=Parameter name PSCT=

in the *Technical Reference*.

Server to **Client Table** (ASST)

Default: [None]

For information, refer to The PSXX=Parameter name PSST=

in the *Technical Reference*.

Connection Failure Action Server fails. (UPCF)

Select the action to take if the connection to the Application

Abort – End the application upgrade and the application.

Continue – End the application upgrade, but continue

executing the application.

Prompt user – Prompt the user for the required action.

Default: Prompt User

Download failure action (UPDF)

Select the action to take if the download from the Application Server fails.

Abort – End the download immediately, and end the

application.

Continue – End the download immediately, but continue with

the application execution.

Prompt user – Prompt the user for the required action.

Default: Prompt User

Application Server Trusted

Yes – Use *Kerberos Authentication* when calling the

CONNECT SERVER Built-In Function.

Connection (ASTC)

No – Do not use *Kerberos Authentication* when calling the CONNECT_SERVER Built-In Function.

Default: No

For information, refer to The PSXX=Parameter name PSTC= in the *Technical Reference*.

8.1.4 SuperServer

Use the LANSA SuperServer installation options to configure your application to automatically connect to a server at run time. You do not need to specify these options if you are using the LANSA SuperServer Built-In Functions in your application.

LANSA SuperServer Built-In Functions include DEFINE_OS_400_SERVER, CONNECT_SERVER and CONNECT_FILE.

If you are using the PSXX parameters, specify the Server Type you wish to connect to.
None - You are not using the PSXX parameters.
RDMLX IBM i - You are connecting to an IBM i server with LANSA enabled for RDMLX. This sets PSTY and DBID to *ANY.
IBM i - You are connecting to an IBM i server. This sets PSTY and DBID to *AS400.
Non-IBM i - You are connecting to a server other than an IBM i (e.g., Windows Server). This sets PSTY and DBID to *OTHER.
Default: [None]
If Non-IBM i is selected you may need to set the values for Server Exceptional Arguments.
Default: [None]
The name of the server you wish to connect to as defined in LANSA Communications Administrator. A corresponding entry should exist in the LANSA Communications Administration table (lroute.dat).
Default: [User]
You do not need to specify this option if the value is the same as the LANSA User.
Default: [None]
If PSPW is only being used to specify the default password for the CONNECT_SERVER Built-In Function do not specify it here. Add the parameter via the X_RUN Arguments dialog.

For details of all the following options, refer to the PSXX= Parameter in the *Technical Reference*.

Server Exceptional Arguments (PSEA)	Default: [None]
Use Commitment Control (PSCC)	Default: [None]
DBCS Capable (PSDB)	Default: [None]
Divert Locks (PSDL)	Default: [None]
Show Please Wait Message (PSWM)	Default: [None]
Server Execution priority (PSEP)	Default: [None]
Client to Server Table (PSCT)	Default: [None]
Server to Client Table (PSST)	Default: [None]
Server Trusted Connection (PSTC)	Yes – Use <i>Kerberos Authentication</i> when calling the CONNECT_SERVER Built-In Function.
	No – Do not use <i>Kerberos Authentication</i> when calling the CONNECT_SERVER Built-In Function. Default: No

8.1.5 Database

These options control whether the package will check for the existence of a database and the connection to the database. The database options are only relevant if the deployed application uses a local database.

The installation and upgrade of Sybase ASA databases is no longer supported by the Deployment Tool.

For more information on the X_RUN database arguments refer to The DBID, DBUT, DBII and DBIT Parameters in the *Technical Reference Guide*.

Setup Database (SUDB)

This value is typically set as Yes.

Web Application Server install, SuperServer Server install and Standalone installs would set this to Yes.

Exceptions are instances where multiple installs access the same shared database. In such a case its up to the installer to check this option in the install dialog for the server and leave it unchecked for the clients.

For example installing a Corporate Application, that is an Application using a shared database, only the first installation of the Application onto a user's PC will require the database to be setup as part of the installation processing. To help end-users install it correctly, the setting should be set to No so that only the Administrator installing it for the first time would need to change the default setting. The Administrator may even install it directly on the Database Server, even though it will not be executed there, just to have a restricted location to install it to.

Its also possible to set it to No and not allow it to be changed in the Local Database Dialog. Then it can only be changed to Yes by running it from the command line. This gives the best protection against inadvertent Database Setup.

To allow the first installation to override the setting of No, the option to *Prompt during Installation* should be selected.

Yes – Install files and file data shipped with the Version or Patch into the nominated database. This will not create the database.

No – The database is not modified.

Default: No

DBMS Type (DBUT)

Select the DBMS type you wish to install into from the List.

No Database

MS SQL Server

Oracle

Sybase ASA

Default: No database.

If this option is set to *No Database* the install processing will automatically set DBII=*NONE and all other database options will be ignored.

Data Source Name (DBII)

The Data Source Name for the database to receive the package.

Default: [Default]

If no database is required leave the **Data Source Name (DBII)** as [Default] and set **DBMS Type (DBUT)** to *No Database*.

If the Data Source Name is set as [Default] the install will attempt to create a Data Source Name using the Application Description. If a Data Source Name already exists with this value this Data Source Name will be used.

Database User (DBUS)

The Database User to be used for connection to database.

Default: [Default]

If the Data Source Name supplied is already defined on the target system, and the Database User is set as *DEFAULT, the Database User will be derived from the existing Data Source Name.

If the Data Source Name supplied is already defined on the target system, and the Database User is set to a value other than *DEFAULT, the Data Source Name definition will be updated with the Database User supplied.

If the Data Source Name supplied does not exist, the package installer will create the Data Source Name using the Database User and Database Password supplied. When the Database User is set to the special value *DEFAULT the User will be set as

follows:

MS SQL Server - sa

Oracle - SYSTEM

Sybase ASA (all versions) – DBA

Database Password (PSWD)

The password to be used for connection to database

Default: [Default]

If the Data Source Name supplied is already defined on the target system, and the Database Password is set as *DEFAULT, the Database Password will be derived from the existing Data Source Name.

If the Data Source Name supplied is already defined on the target system, and the Database Password is set to a value other than *DEFAULT, the Data Source Name definition will be updated with the Database Password supplied.

If the Data Source Name supplied does not exist, the package installer will create the Data Source Name using the Database Password supplied. When the Database Password is set to the special value *DEFAULT the Password will be set as follows:

MS SQL Server – "" (Empty string)

Oracle – MANAGER

Sybase ASA 9 – SQL

Sybase ASA 10 – sql

Sybase ASA 11 – sql

Database Server Name

The name of the database server.

Default: [Default]

If a new Data Source Name is generated by the package installation a value of *DEFAULT will set the Database Server Name to the same value as the Data Source Name.

If a pre-existing Data Source Name is being used a value of *DEFAULT will use the Data Server Name from the existing Data Source Name.

Database Name

The database name to use when installing into an Adaptive Server Anywhere or MS SQL Server database.

Default: [Default]

If a new Data Source Name is generated by the package installation the default will set the Database Name to the same value as the Data Source Name.

If a pre-existing Data Source Name is being used a value of *DEFAULT will use the Database Name from the existing Data Source Name.

Use **Trusted**

This allows the user to specify if Windows Authentication should be used when connecting to a MS SQL Server Database. This Connection option is only relevant for MS SQL Server or Sybase ASA databases.

> **Yes -** Use Windows Authentication. In this case the Database User. and Database Password will be ignored.

No - Do not use Windows Authentication. In this case the Database User and Database Password specified are used to connect to the database.

Default: Yes

8.1.6 Desktop Settings

These settings control the creation of Windows application shortcuts on the target PC's desktop.

Typically, these options are only required for the initial install of an application. For subsequent upgrade packages you should change these options to No.

Create Communication Administration Shortcut

Yes - Create a shortcut to LANSA Communications Extensions.

No - Do not create a shortcut to LANSA Communications Extensions.

Default: Yes

Create ODBC Administration Shortcut

Yes - Create a shortcut to ODBC Administrator.

No - Do not create a shortcut to ODBC Administrator.

8.1.7 Upgrade Options

Delete Previous Target Directory

If this Application has already been installed on the target machine in a different directory, the install process can attempt to automatically remove the previous installation. The Application Code will be used to look up the previous install location in the registry and will attempt to delete the associated folder. This setting can be prompted during the installation.

Yes – Remove the directory where this Application was previously installed.

No – Do not remove the directory where this Application was previously installed.

Default: No

Previous Folder Name **Directory**

If the previous installation is to be removed, indicate the name of **Application** the Previous Application Folder as it was installed. Any desktop icon or program folder matching this name will be deleted during the installation process.

Default: [Application Name]

8.1.8 MSI Installation Dialogs

Many installation options in the Deployment Tool allow for a value to be prompted during installation. Preferably, the end user should not be required to alter the installation settings but this does provide flexibility for the same Version to be installed to different configurations.

Patch installs a limited user interface - the only configuration dialog displayed is the DBMS dialog. Most installation dialogs can only be displayed when installing a Version.

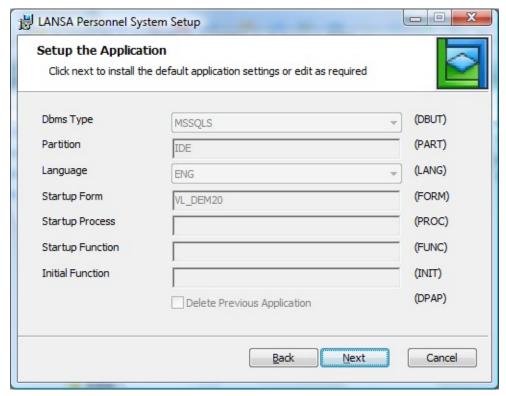
If you wish to use prompting during installation you must ensure that both:

- The Prompt during installation checkbox associated with the option is checked
- The corresponding Installation Dialog is flagged as *Display Yes*.

Important - If the *Prompt during Installation* option is selected but the associated installation dialog is set to *No* the option will NOT be displayed to prompt.

Display Application Dialog Yes - display the Application dialog during installation.

 \boldsymbol{No} - do not display the Application dialog during installation.

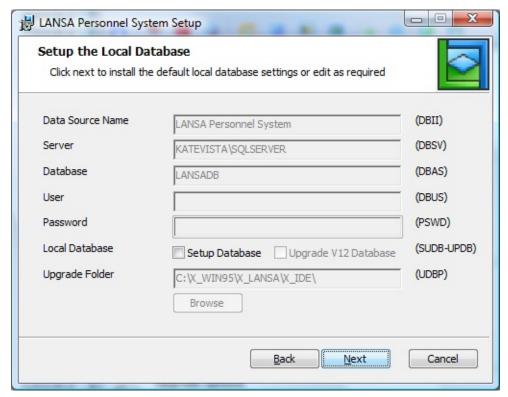


Select of the database system requires the option DBMS Type (DBUT) to be set as Prompt during Installation.

Display DBMS Dialog

Yes - display the DBMS dialog during installation.

 \boldsymbol{No} - do not display the DBMS dialog during installation.

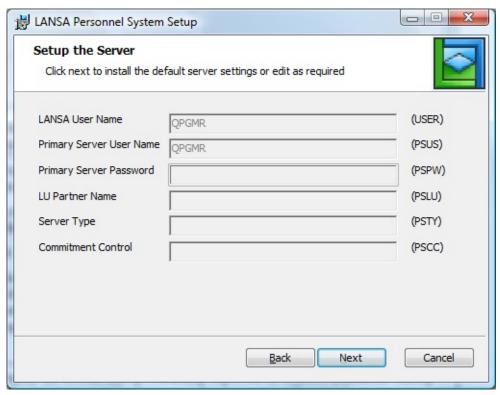


Selection of the Data source requires the option Data Source Name (DBII) to be set as Prompt during Installation.

Display SuperServer Options 1

Yes - display the first dialog containing SuperServer X_RUN connection arguments during installation.

No - do not display the first dialog containing LANSA SuperServer X_RUN connection arguments during installation.

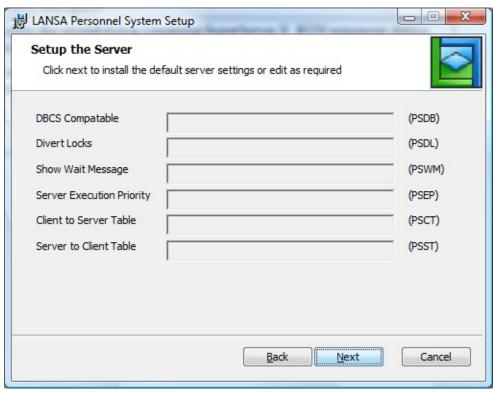


The options on this dialog will only be available to be changed if the SuperServer option Modify SuperServer Options 1 is set to Yes. These options do not have individual checkboxes for Prompt during Installation as their availability as a group is controlled by this option.

Display Options 2

Yes - display the second dialog containing SuperServer X_RUN SuperServer arguments during installation.

> No - do not display the second dialog containing LANSA SuperServer X_RUN arguments during installation.



The options on this dialog will only be available to be changed if the SuperServer option Modify SuperServer Options 2 is set to Yes. These options do not have individual checkboxes for Prompt during Installation as their availability as a group is controlled by this option.

Quiet Install Yes – only show basic dialogs with no waiting during installation. Performs the same function as providing the /qb command line switch to msiexec.exe

> **No** – show the full set of dialogs during installation as configured by other options.

Default: No.

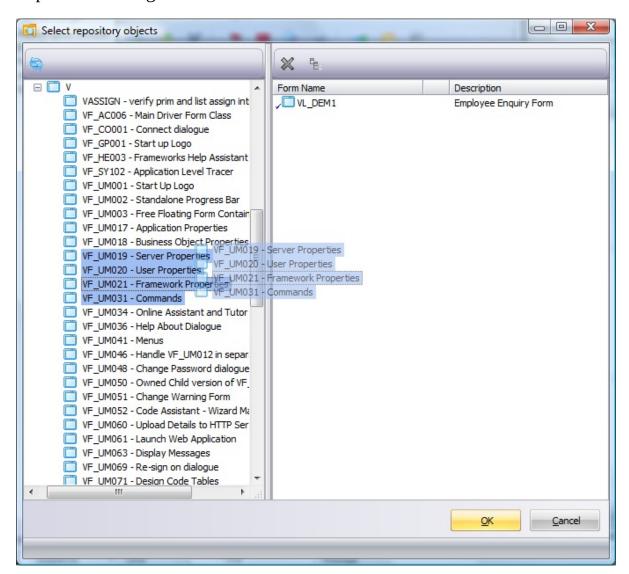
Yes causes an installation to behave exactly like the /qb command line option. Dialogs are displayed but there are no prompts to answer. This is the only setting possible in an MSI or MSP. For more options, the command line has to be used. Using the command line will override whatever setting is chosen for Quiet Install.

This setting allows a JIT install to quietly install an MSI upgrade, like the MSPs do by default. Of course its also allows manual installs to be quiet too.

8.2 Select Repository Objects

To specify the objects to be included in your Application, open the *Select* repository objects dialog by selecting the Repository *Objects* icon from the *Package Maintenance Window* toolbar.

Select the objects for inclusion from the repository tree and drag into the panel on the right hand side.



The object types available for inclusion in a Version or Patch are:

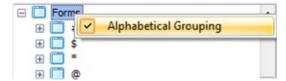
.NET Components	ActiveXs	Bitmaps	Built-in Functions
Cursors	External	Fields	Files

	Resources		
Forms	Icons	Lists	Message Files
Messages	Multilingual Variables	Non-LANSA Objects	Objects from Tasks
Processes & Functions	Reusable Parts	System Variables	Technology Services
Application Templates	Visual Styles	WAMs	Web Components
Weblets			

Different object types have differing requirements, so the 8.2.1 Selected Object Viewer panel changes format to suit each object type. For example, the Process and Function view contains a tree view to show the selected items, whereas Forms are shown as a list.

Selecting Objects

Objects listed in the repository tree can be viewed in alphanumeric groupings (the default) or as a full listing of objects in alphabetically order. Right click on an object type in the repository tree to toggle the alphanumeric grouping on and off.



To select the items to appear in the package, use the mouse and Shift+Ctrl keys to select one or more items in the tree, and then drag and drop them into the selection area on the right. Alternately, double clicking on an item at the lowest level will also select it for inclusion in the package.

Selection is "self expanding". That is, if you select a Process/Function header item, and drag and drop it in the right side of the window, the process and ALL functions within it will be selected.

You can select multiple items at the same tree level. For example, you can select both Components and Processes/Functions. When you drop the selected items on the right, ALL the selected Processes/Functions and Components will be

selected. The Components will be added to the Component list and the Processes and Functions will be added to the Process/Function list.

For more information on object selection refer to 8.2.1 Selected Object Viewer and 8.2.2 Special Considerations for Object Types.

Refresh Repository Objects

Any changes in the repository, specifically the additional or deletion of objects, are not automatically reflected in the Select Repository Objects repository tree. The repository can be manually refreshed using the Refresh option.

Cross References

Cross References enables you to reduce the number of objects explicitly selected for inclusion in a Version or Patch. It is strongly recommended that you define Cross References for your objects.

If you add cross referencing to the selected objects, the build process will automatically include the appropriate related objects used by the selected objects into your package. Refer to Default Cross Reference Settings, 8.2.3 Cross Reference Settings and Objects Included in Package for more information.

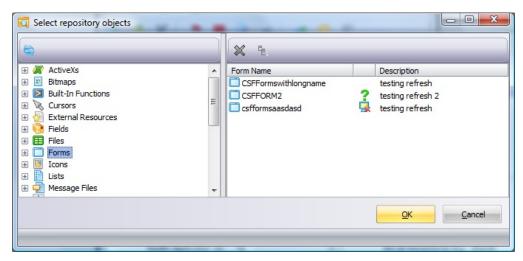
8.2.1 Selected Object Viewer

Objects selected for inclusion in the package can be viewed in the right hand panel on the Select Repository Object dialog.

The tool bar options available in the object viewer are dependent on the type of object being worked with.

Any objects previously included in a package, which no longer exists in the local repository but are known to exist on the master repository are indicated by an \square icon.

Any objects previously included in a package, which are no longer known to exist in the local or master repositories, are indicated by a **(question mark) icon.



The available toolbar options relate to the type of object:

- X This option is available for all object types.
 - To deselect an object, so it will not be included in the package, highlight the object(s) in the selected object list and press the Delete icon.
 - When a Process is deleted, all Functions are also be deleted.
- This option is available for fields, files, forms, processes, functions, reusable parts, system variables, WAMs, and weblets.
 - Select the Cross Reference option in the toolbar to add Cross References for specific package objects. Refer to 8.2.3 Cross Reference Settings for more information.
- This option is only available for files.

You can include the data with a file for deployment and control how this data is loaded with consideration of any existing file data. Highlight the required file(s), and select the *Include File Data* icon in the toolbar the select the appropriate drop down entry to determine how file data will be unloaded and loaded.

Refer to Include and Process File Data for more information.

- This option is only available for files.

 Indicate the library (and subsequently the collection and owner) to be applied when file(s) are deployed.

 Refer to Configure the File Library Substitution for more information.
- This option is only available for non-LANSA objects and Messages. Select *Non-LANSA objects or Messages* in the *Other Objects* list and use the Find option on the toolbar to open an appropriate dialog to search for and select the objects you wish to include in the package. The selected objects will initially be assigned to install in the DFTPATH which is set as the Partition Execute Directory (e.g. /x_ppp/EXECUTE where ppp is the partition identifier) by default.
- This option is only available for non-LANSA objects.

 To install the objects into a different location, highlight the required objects, and select the Set the Install Path option on the toolbar. This will open the *Installation Path Definitions* dialog where you can 8.2.6 Configure the Non-LANSA Object Installation Path.

8.2.2 Special Considerations for Object Types

It is important to understand what objects you need to deploy. This will depend on the environment you are deploying to, and requires some understanding of how objects are bound into a compiled object.

The following list provides some general guidelines for object selection and inclusion. Not all object types are included in this list, only those with special considerations.

Built-In **Functions**

3GL User Defined Built-In Functions will automatically find the associated DLL when the package is built if the Visual LANSA naming convention is followed.

RDML User Defined Built-In Functions can be selected to enable the deployment of the internal definition. The associated function must be added manually to the package.

The associated DLLs can be included in your package by selecting *Deploy Execution Add-ons* in the Settings.

Editor Lists When an Editor List is selected all objects currently included in the Editor List will be included in the package when it is saved.

> When the package is saved, the individual objects will be saved but the selected Editor List is not included as an object. To pick up any changes to the Editor List it must be added to the package again.

🥯 Fields

Field definitions only need to deployed with an application if the application uses the LANSA repository help facilities.

Files

Files need to be included if the application accesses a windows database which is to be set up as part of the application installation.

File data can also be included. Refer to Include and Process File DataInclude and Process File Data.

Messages

Messages are grouped by the Language, Message File, and the first three characters of the message identifier

Use the Search icon A on the toolbar, or double click the Non-

LANSA objects in the tree, to locate and select the required messages.

Messages should only be deployed into other development systems.

Message **Files**

Message files only need to be selected when you are deploying to another development system.

Variables

📤 Multilingual When grouped alphanumerically, Multilingual Variables are grouped by the first alphanumeric character after the *MTXT prefix of the Multilingual Variable name.

> In Visual LANSA, Multilingual Variables are compiled into the application DLLs. There is no need to deploy Multilingual Variables, other than when you are deploying to another development system or to support web applications where multilingual variables are resolved at runtime.

📤 Non-**LANSA Objects**

Non-LANSA objects are Third Party software such as ActiveX controls or parameter files.

There is no definitive set of Non-LANSA objects associated with Visual LANSA so you need to use the Search icon A on the toolbar, or double click the Non-LANSA objects in the tree, to activate a search dialog to locate and select the required objects.

Refer to Configure the Non-LANSA Object Installation Path to understand how to control where the Non-LANSA object will be installed.

Objects from Task(s)

When a task is selected, all objects that have been modified under the task will be immediately included in the package. Removing the task will remove the objects.

Task with Open, Work or Closed status can be added to a package.

Note: When the package is saved the individual objects will be saved but the selected task is not included as an object.

Processes and Function

Functions are available for selection as child items of Processes. If you select a Process, all Functions will be automatically included. If you select just a Function, the Process will be automatically included.

System Variables

Typically, System Variables only need to be included if you are deploying a web application. Process related Variables should be set up as part of Process and Functions cross referencing so that they are automatically included when the package is built.

System Variables related to List Graphical Variables must be included manually.

Technology Services

The standard Technology Services are defined under Technology Provider LANSA.

templates

Application Application templates should only be included in packages for deployment to another LANSA development system. There is no need to include them in the package if you are deploying an execution-only application.

🕽 WAMs

When a WAM is selected the default layout Weblet for the WAM will be automatically added to the package.

Note: When the package is built, any DLLs associated with selected objects will be automatically included in the package unless the Deploying to Development System (specified in Settings) is selected.

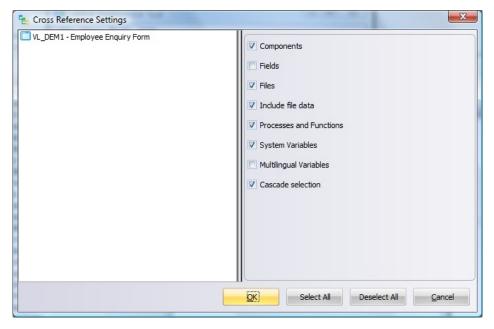
8.2.3 Cross Reference Settings

Cross Reference Settings are available in the Select Repository Objects dialog for object types that may refer to other LANSA objects in their definition. This allows you to select a particular object type and automatically include "related" objects when the package is built.

If Cross References is available for an object type, the Cross Reference

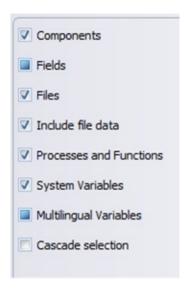
If Cross References is available for an object type, the Cross Reference Icon will be shown on the toolbar for the *Selected Object* panel in the Select Repository Objects dialog.

When the Cross Reference icon is clicked, the *Cross Reference settings* dialog is opened.



The package objects selected to have Cross References applied against them are displayed on the left of the *Cross reference settings* dialog. The available Cross Reference object types are listed on the right of the dialog. Cross Referenced objects are included in the package during build processing and are applied in a top down fashion. That is to say, only objects directly used by the selected package objects can be included in the package using Cross Reference settings.

If the selected objects already have Cross Reference settings applied, the checkboxes will attempt to reflect the settings. Where there is a conflict, that is, a cross reference object type is selected for one object, and the same cross reference object type is not selected for the other object, the Cross Reference checkbox will be grayed.



In addition to the selection of object types associated with the selected object(s) you can:

Include File Data

If Files are available for inclusion as a Cross Reference object, the option to Include File Data will also be available. This option has no effect if the Files option has not been selected.

It is recommended to not use this option unless you have a clear understand of the consequences. Selecting Include File Data will include file data in the first Version as well as any subsequent Versions and Patches and as such may produce undesirable results. If file data should be included in the application it is recommended to explicitly add the file(s) to the package and select the appropriate file data inclusion option. Generally file data is not included with application files and in the case of table files data will only be included in the first version when setting up the database. Subsequent releases of the Application will need to be reviewed to ensure the appropriate file data option is applied.

If you do choose to use this option it is the equivalent of selecting Include Data, Ignore Duplicates in the object selection interface.

Cascade Selection

Cascading Selection causes a flow-on effect such that any object included as a Cross Referenced object will also have its cross referenced object included.

If in the Deployment Tool's Default Cross Reference Settings you selected:

Prompt Options

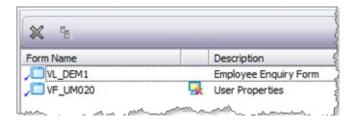
The Cross References dialog will be automatically shown when you press

OK on the Select Repository Objects after adding objects. If you selected to Prompt Options for more than one object type, e.g. Fields and Files, the Cross Reference options will be prompted once per object type. In each Cross Reference dialog press *OK* to accept whatever settings you have chosen and to apply them to all of the objects displayed on the left hand side. Press *Cancel* to leave the dialog without applying Cross References to the objects.

Automatically apply the default options

When object are selected for inclusion in the package in the Select Repository object dialog and you press OK to confirm the selection, Cross Reference information is immediately applied to the object. When you deselect an object (that is, delete it from the selected objects using the Delete icon) any associated Cross References are also removed.

Once Cross References have been applied to an object, it will be shown in the Object Selection dialog with a blue tick.

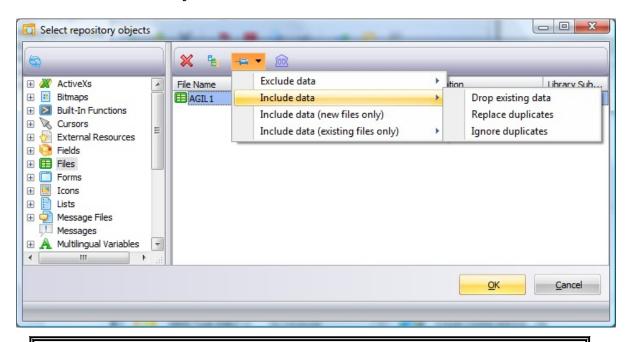


Cross Referenced objects are evaluated when the package is built. Objects included in a package as cross referenced objects can be distinguished from explicitly selected objects so that the next time the package is opened the selected objects are the same.

Refer to Default Cross Reference Settings and Objects Included in Package for more information.

8.2.4 Include and Process File Data

✓ Use the option Include File Data to determine if file data should be included when deploying the file and how to manage any existing file data – if the file already exists – in relation to new file data.



To avoid unnecessarily unloading and reloading file data when installing patches or later versions of an Application where the file definitions have not changed, use the option Exclude data, Reload existing.

Also note, that after installing an MSI, the DAT file is deleted. This means it is not available for a patch to use, unless the patch has changed the table layout or the data is different, in which case the patch will include it. So to avoid complications, with patches, always use the option Exclude data, Reload existing.

It is highly recommended when creating a patch to go through ALL the files you have shipped and change the File Data option to Exclude data, Reload existing.

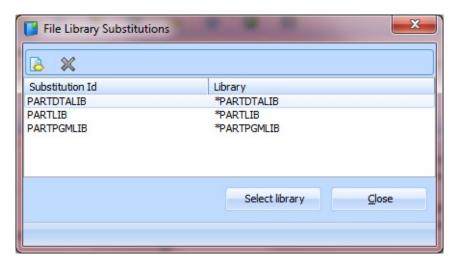
The following processing options are available:

option		
Exclude data, Drop existing	No file data is included in the package. If the file already exists, all data will be removed.	
Exclude data, Reload	exists, all data will be retained.	
existing	When installing patches, if the file definition has not changed, this option should be used to avoid unloading and reloading file data.	
	This is the default option.	
Include data, Drop existing	File data is included in the package. If the file already exists, all data will be removed and package data will be loaded.	
Include data, Ignore duplicates	File data is included in the package. If the file already exists, all data will be retained and package data will be loaded. Any duplicate records in the package data will be ignored.	
Include data, Replace duplicates	File data is included in the package. If the file already exists, all data will be retained and package data will be loaded. Any duplicate records in the package data will replace the existing record.	
Include data (new files	This option is deprecated. It will be removed in the next release.	
only)	File data is included in the package and will be installed if the file does not already exist.	
	If the file already exists in the application no processing of the file data will occur.	
Include data (existing files only), Drop existing	This option is deprecated. It will be removed in the next release.	
	File data is included in the package but will only be installed if the file already exists. Any pre-existing file data is not reloaded.	
Include data	This option is deprecated. It will be removed in the next	

(existing files only), Ignore duplicates	release. File data is included in the package but will only be installed if the file already exists. Any pre-existing file data is reloaded after the new file data and duplicate records in the package data will be ignored.
Include data (existing files only), Replace duplicates	This option is deprecated. It will be removed in the next release. File data is included in the package but will only be installed if the file already exists. Any pre-existing file data is reloaded before the new file data and duplicate records in the package data will replace the existing record.

8.2.5 Configure the File Library Substitution

Use the option Set Library Substitution to assign the library where files are to be installed. The Library Substitution also determines collection and owner properties applied to the table in the target database.



When files are added to a package, a default Library Substitution is derived and assigned based on the file properties and saved Settings.

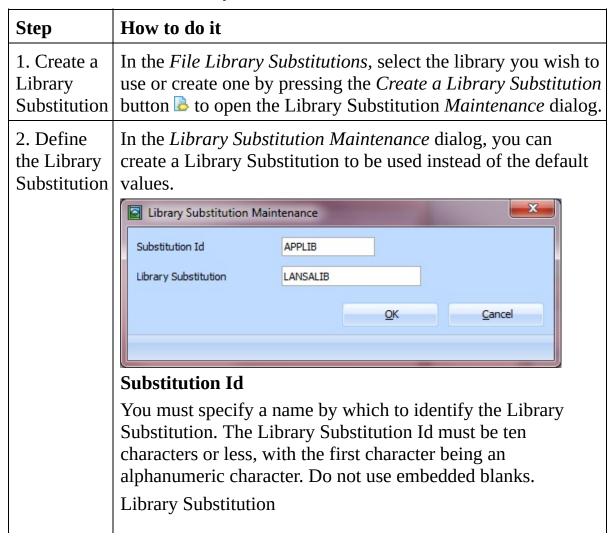
- Files compiled before LANSA V13 and Other files will automatically be assigned PARTLIB as the Library Substitution, indicating the file should be installed at partition level.
- Files compiled with LANSA V13 or later will automatically be assigned PARTDTALIB as the Library Substitution, indicating the file should be installed into the partition Default File Library level.

The following default Library Substitutions are automatically created and cannot be modified, although they can be selected to apply to files if they comply with the restrictions mentioned above:

Library Substitution	Evaluated as
PARTLIB	Files will install at partition level, for example: <sysdir>\X_DEM</sysdir>
	Library, Collection and Owner information is derived from the file OAM and CTD included in the package.
PARTDTALIB	Files will install at partition level, for example:

	<pre><sysdir>\X_DEM<resolved *partdtalib="" of="" value=""> Library, Collection and Owner information is derived from the resolved value of *PARTDTALIB.</resolved></sysdir></pre>
PARTPGMLIB	Files will install at partition level, for example: <sysdir>\X_DEM<resolved *partptalib="" of="" value=""></resolved></sysdir>
	Library, Collection and Owner information is derived from the resolved value of *PARTPGMLIB.

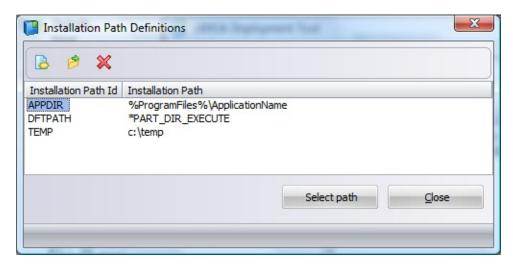
To create an alternative Library Substitution:



	Enter your desired library (schema name) as a free-format entry.
3. Save the Library Substitution	Press <i>OK</i> to save the library substitution. You will be returned to the File Library Substitutions dialog.
4. Modify the Library Substitution	To modify the library substitution, delete the entry and reenter the details as required.

8.2.6 Configure the Non-LANSA Object Installation Path

Use the option to Set the Install Path to set up and configure the target directories where any non-LANSA objects are to be installed.

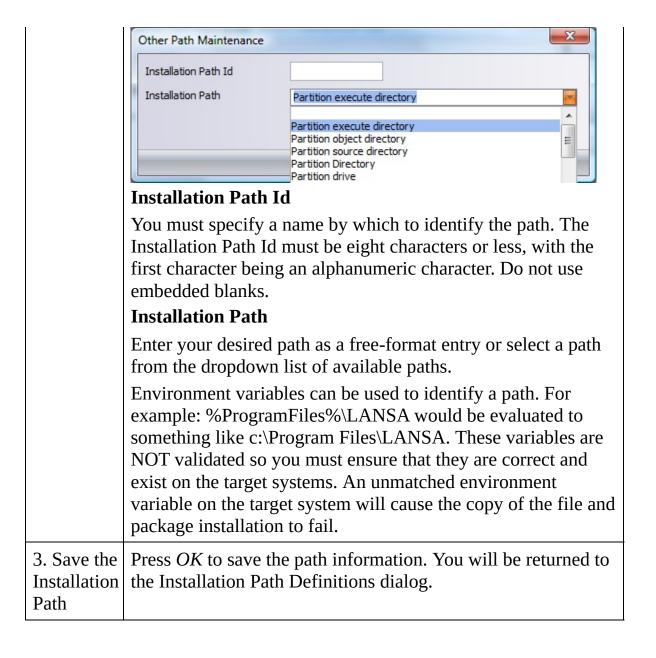


All non-LANSA objects added to the package are automatically set to install into the DFTPATH directory. The DFTPATH corresponds to the partition execution directory where the application is installed. It is recommended not to modify or delete this path.

To assign an alternative installation path, select the non-LANSA object(s) in the Object Viewer then use the Set the Install path option to select (and create or modify) the required installation path.

To create an installation path:

Step	How to do it
1. Create or modify an Installation Path	In the <i>Installation Path Definitions</i> , select the path you wish to use or create one by pressing the <i>Define a new installation path</i> button button to open the <i>Other Path Maintenance</i> dialog.
2. Define the Installation Path	In the <i>Other Path Maintenance</i> dialog, you can create an Installation Path, maintain Installation Paths and select Installation Paths for one or more objects.



The list of predetermined Installation Path Values corresponds with the following system variables:

Installation Path	Evaluated as
Partition Execute Directory	*PART_DIR_EXECUTE
Partition Object Directory	*PART_DIR_OBJECT
Partition Source Directory	*PART_DIR_SOURCE

Partition Directory	*PART_DIR
Partition Drive	*PART_DRIV
System Execute Directory	*SYS_DIR_EXECUTE
System Object Directory	*SYS_DIR_OBJECT
System Source Directory	*SYS_DIR_SOURCE
System Directory	*SYS_DIR
System Drive	*SYS_DRIV
Temp Directory	*TEMP_DIR

To determine the actual directory the corresponding system variable refers to, check the Directories branch of the <u>Installation Details</u> in the relevant LANSA System's *Product Information*, described in the *Visual LANSA User Guide*.

8.3 Settings

Select Settings from the *Package Maintenance Window* toolbar or the Package Edit Menu.

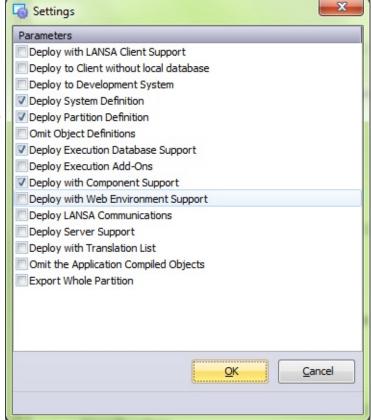
The *Settings* options determine the additional processing that will occur as part during the build processing. Each option selected will include or omit specific parts of the package from the final build.

When deciding which Settings to select, you should consider how the package is to be deployed. Each additional option will add more objects to the package.

If the application is to be installed from a DVD or LAN, the size of the package may not be of great significance.

Upgrade packages will often only need to include changed executable objects (functions and components) as the execution environment will have already been deployed with the initial package.

You should aim to keep **Just In Time upgrades** from a remote server to a minimum size.



Deploy with LANSA Client Support	This option is only required for Applications that are to support a local database. The build processing will execute the special process *CLTEXPORT.

Deploy to Client without local database	This option is recommended for most Client/Server applications. The build processing will execute the special process *SYSEXPORT. Select this option to include the *.XQ* files required for the application to execute. The .XQ* files are used in place of a local DBMS for read-only dictionary/repository access such as prompting, display help text, etc. Any PC that executes an application using DBID=*ANY, DBID=*AS400, DBID=*OTHER, DBID=*NONE or DBII=*NONE needs to have the special .XQ* files created or installed on it, or available to it via a connected server disk drive. Refer to The *.XQ files in Technical Reference for more
	information.
Deploy to Development System	This option is available so developers can export source code. Select this option to include the complete repository definitions of the selected objects. This option is not required for packages that contain executable applications but is intended as a means to distribute LANSA internal object definitions, including function and component source, field and file definitions, task tracking details, cross reference information and other LANSA objects. Selection of this option will cause the MSI generation to be bypassed in the build processing. There will be no MSI or MSP file produced. Definitions must be imported from the\X_PKGWRK\1.0.0\Build\internal directory. Refer to LANSA Import For more information.
Deploy System Definition	This will cause the Export of the LANSA internal database to include the system definition table (LX_F96) and other system related details. The information in these tables is required when executing

	an application. This option should be chosen if you expect the package to be installed into a new database, or to a new system.
Deploy Partition Definition	This will cause the Export of the LANSA internal database to include the partition definition table (LX_F46), the partition language table (LX_F60) and other partition related details. The information in these tables is required when executing an application. This option should be chosen if you expect the Application to be installed into a new database, or to a
Omit Object Definitions	new system. This option will bypass the Export of the LANSA internal database definitions for the selected objects (including LX_F02, LX_F03, LX_F04 and LX_F62). It will allow you to ship the package without the internal definitions.
	This should NOT be selected when Deploy to Development is selected or if deploying files. Object Definitions are required when deploying to a development system, when deploying files or if deploying to a system where you intend to use the database for accessing the internal database definitions required at execution time.
Deploy Execution Database Support	A copy of the LANSA database support image is included in the package. This setting is required if your application needs access to a local database. Once an application is deployed, there is no need to include the execution database support in subsequent "upgrade" packages, except for major LANSA version upgrades.
Deploy Execution Addons	MAPI support is to be included in the package. Typically this is only included if your application incorporate email functionality. Once an application is deployed, there is no need to

	include the execution add-ons in subsequent "upgrade" packages, except for major LANSA version upgrades.	
Deploy with Component Support	A copy of the component support execution environment image is included in the package and installed when the package is installed. This is a requirement for all applications that are to execute Visual LANSA components.	
	Once an application is deployed, there is no need to include the component support in subsequent "upgrade" packages, except for major LANSA version upgrades.	
Deploy with Web Environment Support	A copy of the LANSA for the Web on Windows execution environment installation images are included in the package and installed when the package is installed.	
	This is a requirement for all applications that are to execute LANSA Web applications.	
Deploy LANSA Communications	A copy of the LANSA Communications Extensions installation image is included in the package and is installed when the package is installed. This includes all the support required for a LANSA SuperServer client application.	
	This is a requirement for all Client/Server applications and must be included if the <i>Deploy Server Support</i> option on this dialog is selected.	
	Once an application is deployed, there is no need to include the LANSA communications in subsequent "upgrade" packages, except for major LANSA version upgrades.	
Deploy Server Support	A copy of the LANSA communications server support installation image is included in the package and is installed when the package is installed.	
	This option only needs to be selected if the package is to be installed on a server system that will be receiving LANSA SuperServer connections from remote clients. The <i>Deploy LANSA Communications</i> option in this	

Deploy with Translation List	dialog must also be selected if this option is selected. Once an application is deployed, there is no need to include the server support in subsequent "upgrade" packages, except for major LANSA version upgrades. This is a specialized option to facilitate the translation of LANSA object related information and is typically used during the application development cycle to distribute the object information to translators. When this option is selected, the LANSA internal definitions for the selected objects will be stored in flat files (i.e. TL*.DAT files) and supplied with the application for translation. A translation package must be installed into an existing LANSA system where the translator is able to open the translation dialog to complete the translations. Once translated, the information should be zipped and returned to the developer for loading back into their development system using the <i>Import Translations</i> command on the Tools menu in Visual LANSA. Refer to Translating a LANSA-developed Application
Omit the Application Compiled Objects Export Whole Partition	If this option is selected, the package builder will not copy the executables for the selected LANSA objects to the package. The build process will fail if this option is selected and the package includes any files with data. Note: This setting should only be used when deploying to another development system. If the Deploy to Development System option in this dialog is selected with this option, when the internal data is loaded, all files included will automatically have their library updated to that of the target partition's data library. This option will cause the Export of the LANSA internal database to include the definitions for everything in the partition regardless of any object

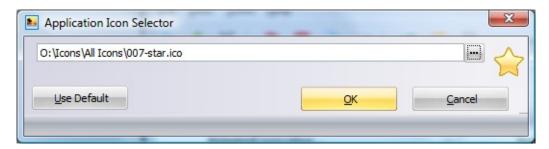
Note: This option will only cause the definitions of all
objects to be deployed. Objects must be selected if you
wish to deploy them for execution.

☆ ImportantLANSA Execution Environment and any installed EPCs will be automatically included in all generated MSI and MSP files.

8.4 Application Icon

The *Application Icon Selector* allows you to choose the icon to be used in the Windows Package Maintenance Window and for the desktop shortcut on the computer where the Application is installed.

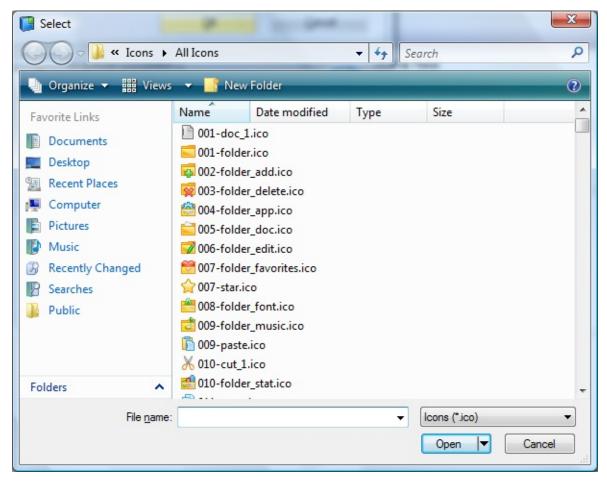
► To open the *Application Icon Selector*, select the *Icon* tool from the *Package Maintenance Window* toolbar.



Once you have selected an icon, it will be displayed to the right of this dialog as shown in this screen picture.

Press *Use Default* to select the default LANSA icon.

You can enter the path and icon name or prompt using the button to open a common file dialog to search the icon to be used in the application.

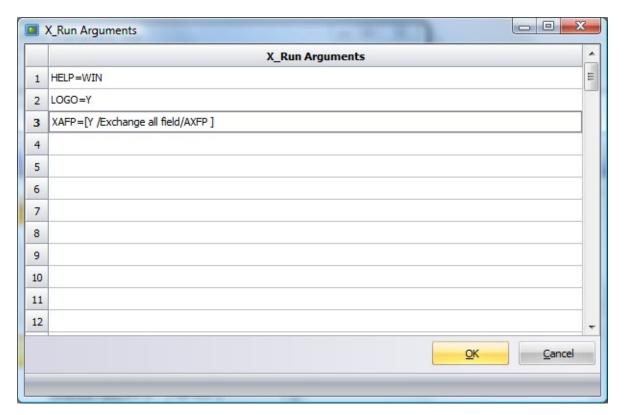


Select the icon that you required and press *Open* to return to the *Application Icon Selector* dialog. The icon that you have selected will be displayed to the right of the path.

8.5 X_RUN Arguments

When a package is built, the X_RUN arguments from the *Installation Options* will be combined into a start file. The options in the *Package Maintenance Window* cover the most commonly used arguments. However, there are other settings that you may want to include to change the behavior of your application at run time.

You can enter these other settings using the *X_RUN* arguments view dialog box. Open this dialog by selecting the *X_RUN* icon from the *Package Maintenance Window* toolbar.



When adding a new argument, it must be entered in the same format as used for an X START file.

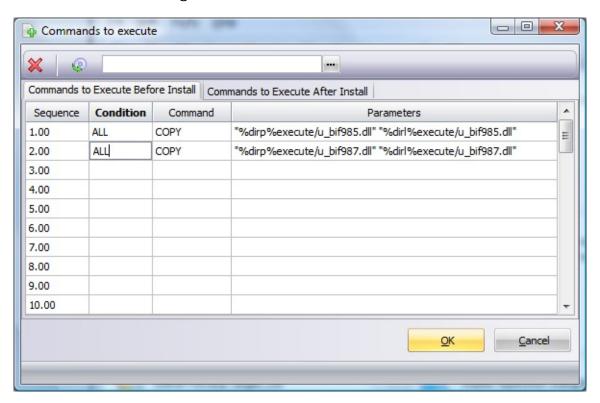
Arguments entered here will be added into the X_START file AFTER the automatically created entries that are prompted, in alphabetical order. Therefore if a promptable parameter is used the prompt will also be after the automatically prompted parameters.

For more information, refer to Using X_START as a Front End to X_RUN in the Technical Reference.

8.6 Commands to execute

Any commands entered allow you to execute operating system level commands before or after a package is installed on a target machine.

To open the Commands to execute dialog, select the *Commands to execute* icon from the *Package Maintenance Window* toolbar.



Separate tabs are created for entry of commands to execute before package installation and after package installation.

Select the *Commands to run before Install* tab to enter the commands you wish to execute *before* the install.

Select the *Commands to run after Install* tab to enter the commands you wish to execute *after* the install.

Enter/amend the commands as you require in the relevant tab.

Press the *OK* button when the command entry is completed. The commands are updated when the package is saved.

- Clear all commands from the currently open tab.
- Import commands into the currently opened tab from the tab delimited file nominated in the associated text box.

Important - You are responsible for ensuring that the package installs correctly, and that the commands executed before and after the LANSA installation processing execute correctly and in the sequence you require.

The Conditions and Commands available for use are provided in these lists:

- 8.6.1 Conditions to Control the Execution of a Command
- 8.6.2 Valid Commands and Parameters
- 8.6.3 Valid Substitution Variables for Command Parameters

Tips when using commands to execute

- Directory and path type substitution variables that contain imbedded blanks when resolved will cause the execution of the LANSA Import to fail. To correct this issue the value should be enclosed in double quotes. For example, "%dirp%execute".
- If you include the command MD (make directory) all intermediary directories must exist. For example if you intend to create a directory c:\Program Files\MYAPP\other objects when currently only c:\Program Files can be assumed to exist you will need to include the commands:

MD "c:\Program Files\MYAPP"

MD "c:\Program Files\MYAPP\other objects"

8.6.1 Conditions to Control the Execution of a Command

Condition Meaning		Negative Condition	Comments	
	*	Comment Line	Not available	Line ignored
	ALL	Always true	Not available	
	OSENG	Operating System language is English	NOSENG	Win 32 environments only
	OSFRA	Operating System language is French	NOSFRA	Win 32 environments only
	OSJPN	Operating System language is Japanese	NOSJPN	Win 32 environments only
	UNIX	Operating system is UNIX	NUNIX	
	WIN	Operating system is Windows (any)	NWIN	
	WIN32	Operating System is Windows 32 bit	NWIN32	
	WIN95	Operating System is Windows 9X/98	NWIN95	
	WINNT	Operating System is Windows NT	NWINNT	

8.6.2 Valid Commands and Parameters

Command	Parameter(s)	Comments
ABORT	Message Text	Aborts execution showing the message specified.
ATTRIB	File Name	
	Attribute	Changes the attributes of the specified file. Specify the attribute as +r or -r only.
CLOSE	None	Close the open output file.
COPY	From File	Copies the specified file.
	To File	
COPYGROUP	File Suffix	Copies a group of files by generic name and
	Target Directory	type to a target directory.
DEL	File Name	Deletes the specified file.
EXECFATAL	.EXE File Name	Execute the specified file with the specified parameters. This is the same as the
	Parameters	EXECUTE command except that if an error occurs while running the application, it is counted as a fatal error. (EXECUTE counts it as a warning). Errors are either errors in launching the application such as 'File Not Found' or exit codes returned by the application. (For Windows platform details, refer to Microsoft documentation on the CreateProcess and GetExitCodeProcess functions.)
EXECUTE	.EXE File Name	Execute the specified file with the specified parameters.
	Parameters	

FNEXTEXPR	File Name	As per NEXTEXPR, but import is forced with a question.
LAUNCH Note: This command is reserved for system use.	Process Name	Starts a LANSA process and/or function executing.
	Function Name	
LAUNCHFORM	Form Name	Starts a LANSA form executing.
Note: This command is reserved for system use.		
MD	Directory Name	Makes the specified directory.
MESSAGE	Message Text	Shows the specified message in the log.
NEXTEXPR	File Name	Queues up a subsequent import.
NOPARTWARN	None	Turns off different partition warning message.
Open	File Name	Opens the specified file for output.
QUESTIONn	Text of question	Asks a question to in a message box (n in range 1 to 9).
RD	Directory Name	Removes the specified directory.
ROLLSET	*YES	Sets a flag to roll sets of records when they
	*NO	are imported. Although implemented to support the default Web Pages, it may be used for other file types.
SCANREP Note: This command is reserved for system use.	Source Character Target Character	Converts characters in the imported data from the Source Character to the Target Character. The following special values are supported for the Target Character: *HASH_SIGN_CHAR, *AT_SIGN_CHAR, *DOLLAR_SIGN_CHAR.

SYSTEM	System Command	Executes the specified system command.
USEPART	*SOURCE	Sets the partition to be the partition from which the export came.
	*INITIAL	Sets the partition to be the initial partition specified by the user when starting the import. *INITIAL is the default.
VLDATA Note: This command is reserved for system use.	*YES *NO	Indicates that this data was exported from a Visual LANSA system. This is generated if necessary when using the Deployment Tool or the EXPORT_OBJECTS Built-In Function. * NO is the default.
WRITE	Data	Writes the specified data to the output file.

8.6.3 Valid Substitution Variables for Command Parameters

Variable	Comment	
%PART%	Current Partition Identifier	
%LANG%	Current Language Identifier	
%DIRL%	Current LANSA root Directory	
%DIRP%	Current Partition Directory	
%TEMP%	Current Temporary Directory	
%PKGD%	Current Package Directory	
%EXPR%	File being import from	
%SOSD%	Current operating system prefix (Linux only)	
%DBID%	Current DBID value	
%DBII%	Current DBII value	
%DBIT%	Current DBIT value	
%DBUT%	Current DBUT value	
%DBUS%	Current DBUS value	
%PSWD%	Current PSWD value	
%YQUn%	Command line is ignored unless answer to QUESTIONn was yes	
%NQUn%	Command line is ignored unless answer to QUESTIONn was no	
%EXPD%	Original EXPR parameter used in import	
%LFRA%	Command line is ignored unless LANSA language code is FRA (French)	
%LJPN%	Command line is ignored unless LANSA language code is JPN (Japanese)	
%LOTH%	Command line is ignored unless LANSA language code is not FRA or JPN	

- %MULT% Command line is ignored unless the current partition is multilingual
- %MONO% Command line is ignored unless the current partition is monolingual
- %USER% Current USER value

8.7 Languages

The *Languages* dialog allows you to select the installation language(s) to be used when a package is installed. To open the *Languages* dialog select the *Language* icon from the *Package Maintenance Window* toolbar.

The current language is used as the default language and as such is preselected.

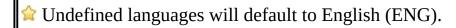
The current partition languages will be displayed in the *Languages* dialog with the current language from the Visual LANSA log in screen selected by default.

When the installation package is built by the Deployment Tool, an MSI or MSP file, depending on whether this package is a full application version or a patch, will be created for every language selected via the *Languages* dialog. The resulting Windows Installer files are named by concatenating the application code with the version number and the "culture code" derived from these language codes.

For example, selecting English, French and Japanese for application D20A, version 1.22.33 will result in the following build output:

D20A_v1.22.33_en-us.msi D20A_v1.22.33_fr-fr.msi D20A_v1.22.33_ja-jp.msi

This will allow the installation of an English, French and Japanese application to be performed by the end user. Unlike different versions of an application that cannot exist as side-by-side installations, language variants may be installed on the same computer at the same time.



8.7.1 Custom Language Definition

The language code pairs used to generate the MSI and MSP file names are mapped in $\langle sysdir \rangle X_Apps X_Wix Content Languages.xml$

To support additional languages modify the Languages.xml file by adding an entry for the required language in the format:

```
<Language Id="WXYZ" Culture="wx-yz" Locale="2052" CodePage="950"/>
```

In addition you will need to define an appropriate WIX localization file in ... $X_AppsX_WixSource$, for example, file wx-yz.wxl.

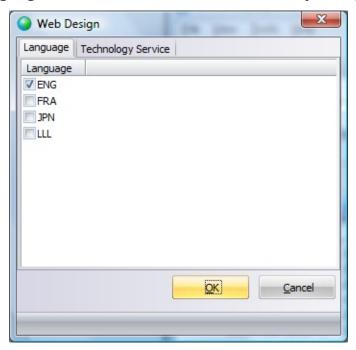
8.8 Web Designs

Indicate the Languages and Technology Services to be shipped with any WAM or weblet included in the Application.

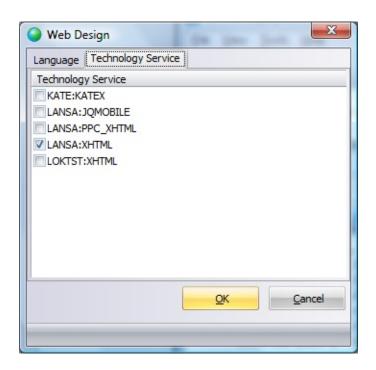
If WAMs and weblets have been developed with multiple Web Designs to support different languages and/or technology services, use this feature to select the web designs to be included in the current deployment.

By default only the web design for the current language and technology service will be selected.

The Current Language is nominated on the screen when you log on to LANSA.



The Current Technology Service defaults to LANSA XHTML but can be changed on the Web menu in the LANSA Editor.



8.9 Files with Special Processing

The Files with Special Processing dialog opens from the Package Maintenance Window toolbar.

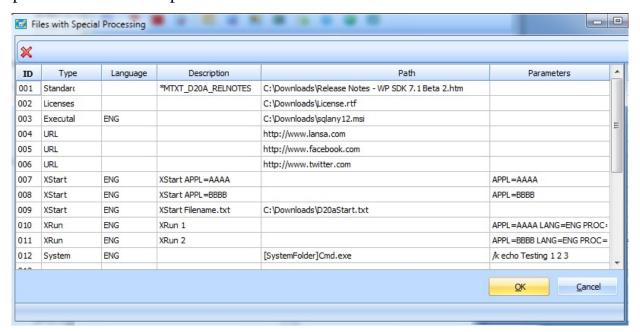
Special Files are files to be installed in predefined locations with specific processing per file *Type*.

Files can be delivered using the *Files with Special Processing* dialog. As well as delivering these files, the *Special Processing* associated with the Type nominated determines how it is processed during installation. For example, shortcuts, X_START files or the definition of an End User License Agreement can be delivered.

To deliver files which do not require special processing use the *Object Selection* dialog and include these files as Non-LANSA objects, or as External Resources if they are defined in the LANSA database.

Shortcuts to files such as .doc and .rtf rely on Windows file associations to work, therefore the applications that load these files must be installed separately.

All shortcuts will be created in the Application's program folder and not placed on the end user's desktop. Only the default Application shortcut is automatically placed on the desktop.



ID Required.

Sequential numbers are automatically generated but this can be

re-sequenced as required.

Type Required.

The [Shortcut] Type can be selected from drop down list. The supported types are:

Standard, Licenses, Readme, URL, XRun, XStart,

Executable, System, WebUsers, ActiveX

Shortcuts to existing executables can be created using the *System type*.

Shortcuts to web pages can be created using the *URL type*.

Licenses supplies a text file to be used as the License Agreement when running the MSI File. This will replace the file supplied in ...\X Apps\X Wix\Content\license.rtf

Refer to End-User License Agreement Dialog for more information.

XRun will run X_Run.exe with any Parameters provided.

XStart will run X_Start.exe using the original start file as the basis, with overrides applied as supplied in the Parameters.

WebUser indicates the DC_W08.dat file.

There are three ways to execute non-LANSA objects:

Standard: Ship the object and Open it. (Shortcut text has 'Open'). This is most appropriate for non-executables, like a spreadsheet. The application that opens the object must already be installed.

Executable: Ship the object and execute it with parameters. (Shortcut text has 'Start)

System: Execute an object which is already installed with parameters. (Shortcut text has 'Start')

All three, *Standard*, *Executable* and *System*, can be used with any file extension that has a registered default program on the installed machine.

Language Optional.

Leave blank for all languages supported by the Application or select a Partition Language from the drop down list. If a language is entered, ensure this is also selected in the Languages.

Files may be targeted at specific language installations.

For all languages, leave the Language column empty (meaning all languages) and use a *Multilingual Variable* to provide an appropriate description if multiple languages are supported.

For a single language, select the specific language from the drop down list of Partition Languages. Ensure the selected language is a selected as a supported package language. Enter a description appropriate for this language or use a multilingual variable.

Description Optional.

Multilingual variables can be selected to provide a description appropriate to each language process.

If no Description is entered, the filename will be used.

Path Optional.

The target path where special files will be installed.

The System Type may use a Path with a sub-directory, for example:

[PersonalFolder]MySubDir\MyThesis.docx

The following substitution variables are supplied by MS Windows and may be specified in the Path surrounded by square brackets, for example [SystemFolder].

Substitution variable Expanded Path

AdminToolsFolder C:\ProgramData\Microsoft\Windows\Start

Menu\Programs\Administrative Tools\

AppDataFolder C:\Users\Installer.NYB\AppData\Roaming\

 $Common App Data Folder \ C: \ \ Program Data \\ \\$

C:\Program Files\Common Files\

FontsFolder C:\Windows\Fonts\

LocalAppDataFolder C:\Users\Installer.NYB\AppData\Local\

MyPicturesFolder C:\Users\Installer.NYB\Pictures\

NetHoodFolder C:\Users\Installer.NYB\AppData\Roaming\Micro

Shortcuts\

PersonalFolder C:\Users\Installer.NYB\Documents\

PrintHoodFolder C:\Users\Installer.NYB\AppData\Roaming\Micrc

Shortcuts\

ProgramFilesFolder C:\Program Files\

RecentFolder C:\Users\Installer.NYB\AppData\Roaming\Micrc

SendToFolder C:\Users\Installer.NYB\AppData\Roaming\Micrc

StartMenuFolder C:\ProgramData\Microsoft\Windows\Start Menu\

StartupFolder C:\ProgramData\Microsoft\Windows\Start Menu\

System16Folder C:\Windows\system\

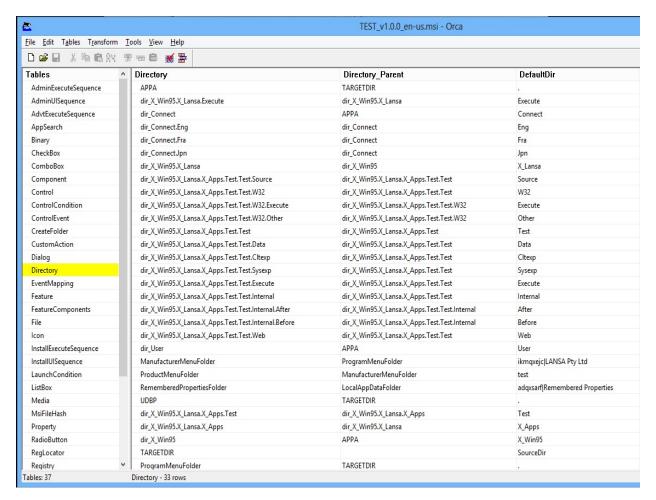
SystemFolder C:\Windows\system32\

TempFolder C:\Temp\

TemplateFolder C:\ProgramData\Microsoft\Windows\Templates\

WindowsFolder C:\Windows\

Any path that exists in the MSI Directory table may be used in the same way. You can use Orca on an MSI (right click Edit with Orca) to see the Directory table. Following is an example of the Directory table:



For example, to reference myapp.exe in the x_win95\x_lansa\execute directory use [dir_X_Win95.X_Lansa.Execute]myapp.exe

To obtain Orca, install the Windows SDK and run Orca-x86_en-us.msi. In version 8.0 of the SDK you will find it here: C:\Program Files (x86)\Windows Kits\8.0\bin\x86

Parameters Optional.

Depending on the File Type selected, parameters can be provided to execute with file processing. Parameters are only applicable for *XStart*, *XRun*, *Executable* or *System* File Types.

9. Deploy LANSA Communications

For a general understanding of LANSA Communications refer to:

• What are LANSA Communications extensions? in the *LANSA Communications Setup Guide*.

Also refer to these topics in this guide:

- Include Host Route and Listener Tables
- For package settings and installation options relating to LANSA Communications, go to LANSA Communications Package Settings & Options.

Translations for LANSA Communications Administrator are only required if you are working in a language other than English, French or Japanese and want to review or modify the lroute.dat or listener.dat files using the LANSA Communications Administrator interface running in your preferred language.

9.1 Include Host Route and Listener Tables

The LANSA Communications Administrator is incorporated into the Deployment Tool to allow the creation and maintenance of a communication files specific to an Application. The communication files are the Host Route Table (lroute.dat) and Listener Table (listen.dat).

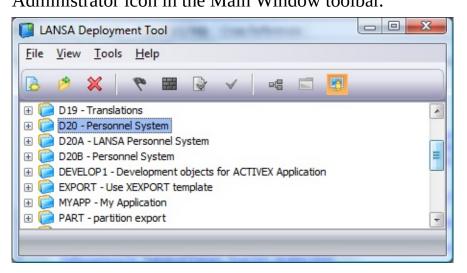
When the Communication Administrator is first opened against an Application a default set of communication files is created. These files can then be modified as required to support the Application.

To include the Communication files:

Step How to do it

1. Select To include or change Communication files, select the Communications Application in the tree view then press the Communications Administrator Administrator icon in the Main Window toolbar.

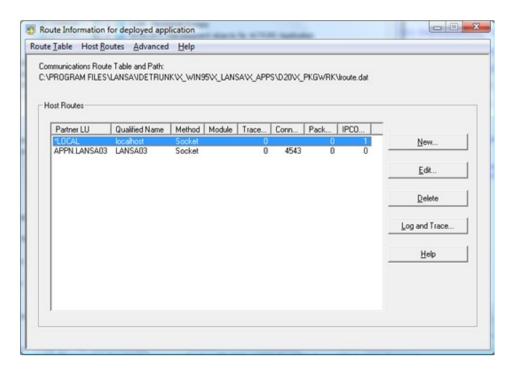
for an application



2. Set up the Host Route entries

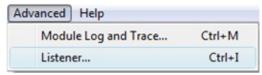
The Route Information for deployed application dialog box will open to display a copy of the host route table lroute.dat from the current LANSA system. This version of the host route table is saved into the Application work folder.

Change, remove and add host route entries in the local communication files as required to support the deployed Application. Closing the dialog box saves any changes.



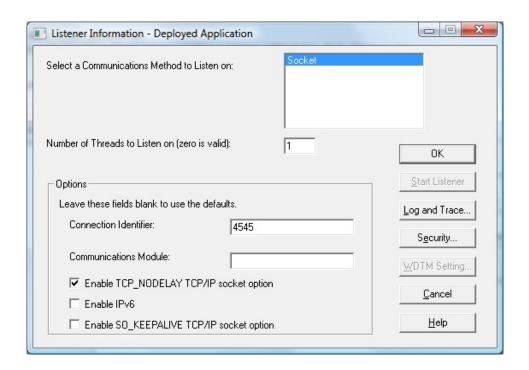
3. Set up the listener

If you are Deploying Server Support or installing a web application you may also need to configure the Listener port. On the Route Information for deployed application dialog select the Advanced menu option Listener...



This will create a default listener table listen.dat. If necessary change the listener information as required to support the deployed application. Closing the dialog box saves any changes.

This version of the listener table is saved into the Application work folder along with the host route table.



4. Set appropriate and options

Including the Communication files in an Application does NOT automatically mean they will be installed with a Version Package settings or Patch. The appropriate Settings and Installation options must also be selected.

> Refer to 9.2 LANSA Communications Package Settings & Options for more information.

9.2 LANSA Communications Package Settings & Options

The following package settings should be reviewed when you are deploying LANSA Communications:

Deploy LANSA Communications	A copy of the LANSA Communications Extensions installation image is included in the package and is installed when the package is installed. This option must be selected to deploy the lroute.dat file. Refer to Deploy LANSA Communications.
Deploy Server Support	A copy of the LANSA Server Support installation image is included in the package and installed when the package is installed.

Communications

Install Listener	Yes – Listener will be installed
	No – Listener will not be installed
	Default: No

Desktop Settings

Create Communications Administration shortcut	Yes – Create a shortcut to LANSA Communications Extensions
	No – Do not create a shortcut to LANSA Communications Extensions Default: Yes

10. Deploy Client / Server Applications

A myriad of Client / Server configurations can be implemented. Before you start deploying any Client / Server application it is essential to understand:

- What pieces of the application belong on the server and what pieces belong on each client.
- What are the requirements of the runtime environment on the respective client and server.
- How the client is intended to communicate with the server.
- How are updates to the client and server going to be managed.
- If you have appropriate LANSA licenses to install.
- Scenario? Network Client? Web Application?

Some of the common Client / Server scenarios that can be implemented are:

- 10.2 SuperServer Application using Built-In Functions to connect to server
- 10.4 SuperServer Application connecting to an IBM i server
- 11.2 Data Application Server with Windows Application Database
- 10.5 Network Client Installation

10.1 Client / Server Deployment Considerations

- If the Client / Server application includes the package settings Deploy system definition or Deploy partition definition ensure the Visual LANSA system where you are creating the application has the appropriate system and partition settings as required in the runtime environment.
- If your application uses SuperServer to connect to a remote system it is recommended to embed the SuperServer Built-In Functions in the application logic rather than using the SuperServer options supplied with the Deployment Tool. This provides a consistent interface which is encapsulated and controlled within the application code.
- Ensure Communication Administration file (lroute.dat) and listener (listen.dat) are updated with correct information to connect to server
- Ensure the option to Install Listener is selected when listener is required
- When installing the server side of an application the setting Database Setup (SUDB) should be set to ensure the database is modified. Conversely when installing the client the setting Database Setup (SUDB) should not be set and only runtime objects should be installed.

10.2 SuperServer Application using Built-In Functions to connect to server

In this scenario the database is installed on the server and the clients are installed as "fat" clients with the application DLLs installed locally. Connection to the server is established using SuperServer Built-In Functions.

Using SuperServer Built-In Functions is the recommended approach for establishing a connection to a remote system. This decision is made during the application development.

To support this scenario:

1. The server portion of the deployed application will depend on the type of server used.

For guidelines on setting up the server application, including the system variables, files (including message files) and their OAMs refer to, as applicable:

- 10.2 SuperServer Application using Built-In Functions to connect to server
- 10.4 SuperServer Application connecting to an IBM i server
- 11.2 Data Application Server with Windows Application Database.
- 2. Executable for the deployed application will be installed on each client including the application's forms, processes and functions. Database accesses will be directed to the server.

Client Package

The client package can be based on template XCLTBIF.

The following Settings should be considered and reviewed:

- Deploy to Client without local database
- Deploy System definition
- Deploy Partition definition
- Omit Object Definitions
- Deploy with Component Support
- Deploy LANSA communications

Omit any unnecessary options to minimize the package size. Most of these options only need to be included in the initial package or after LANSA software

upgrades have been applied and need to be deployed.

The following Options and details should also be considered and reviewed:

- Process/Function/Form to Execute. Specify the application executable that is to be used to start the application once installed.
- Install Listener.
- The Communication Administration file (lroute.dat) and listener (listen.dat) will need to be included to facilitate connection to the data server.

Note: The DBMS Type is typically set as NONE (that is, no local database).

10.3 SuperServer Application connecting to a Windows or Linux Server

In this scenario the database is installed on the server and the clients are installed as "fat" clients with the application DLLs installed locally.

To support this scenario create two Deployment Tool applications:

- 1. The first application will be installed on the server and includes system variables, files (including message files) and their OAMs.
- 2. The second application must be installed on each client and includes the application's forms, processes and functions.

Server Package

A windows server package can be based on template XSRVOTH.

The following Settings should be considered and reviewed:

- Deploy System definition
- Deploy Partition definition
- Deploy Execution Database Support
- Deploy Execution Add-ons
- Deploy with Component Support
- Deploy LANSA Communications
- Deploy Server Support

The following Options and details should also be considered and reviewed:

- Install Listener.
- Communication Administration file (lroute.dat) and listener (listen.dat) will need to be included to facilitate connection to the data server.
- DBMS Type (DBUT). An appropriate server database type should be nominated, for example, Oracle or MS SQL Server.
- Database Source Name (DBII).

Client Package

Using SuperServer Built-In Functions is the recommended approach for establishing a connection to a remote system. For details on installing a Client application which utilizes SuperServer Built-In Functions refer to Superserver Application using Built-In Functions to connect to server.

The client package can be based on template XCLTOTH and requires the appropriate X_RUN parameters to be provided to connect to a server.

The following Settings should be considered and reviewed:

- Deploy to Client without local database
- Deploy System definition
- Deploy Partition definition
- Deploy Execution Add-ons
- Deploy with Component Support
- Deploy LANSA Communications

The following Options and details should also be considered and reviewed:

- LANSA User
- Process/Function/Form to Execute Specify the executable to be used to start the application once installed
- Install Listener
- The Communication Administration file (lroute.dat) and listener (listen.dat) will need to be included to facilitate connection to the data server.
- SuperServer options:
- Server Type (DBID)
- Server Name (PSLU)
- User for Server (PSUS)
- Password for Server (PSPW)
- Divert Locks (PSDL)
- Server Exception Arguments (PSEA) (for example DBID, DBII, DBIT, DBUT, DBUS, PSWD, RPTH, TPTH)
- DBMS Type (DBUT). Typically set as NONE (that is, no local database).

10.4 SuperServer Application connecting to an IBM i server

In this scenario the database is installed on an IBM i server and the clients are installed as "fat" clients with the application DLLs installed locally.

To support this scenario create a single Deployment Tool application to be installed on each client. This will include the application's forms, processes and functions much the same as when connecting to a Windows or Linux client.

The IBM i portion of the application must be deployed using the LANSA for iSeries Export and Import facilities. This will include the system variables, files (including message files) and their OAMs.

Client Package

Using SuperServer Built-In Functions is the recommended approach for establishing a connection to a remote system. For details on installing a Client application which utilizes SuperServer Built-In Functions refer to Superserver Application using Built-In Functions to connect to server.

The client package can be based on template XCLT400 or XCLT400X and requires the appropriate X_RUN parameters to be provided to connect to an appropriate LANSA for iSeries server and partition.

The following settings should be considered and reviewed:

- Deploy to Client without local database
- Deploy System definition
- Deploy Partition definition
- Omit Object Definitions
- Deploy with Component Support
- Deploy LANSA Communications

The following Options and details should also be considered and reviewed:

- LANSA User
- Process/Function/Form to Execute Specify the executable to be used to start the application once installed
- Install Listener
- Communication Administration file (lroute.dat) and listener (listen.dat) will need to be included to facilitate connection to the data server.

- DBMS Type. Typically set as NONE (that is, no local database).
- SuperServer options:
- Server Type (DBID)
- Server Name (PSLU)
- User for Server (PSUS)
- Password for Server (PSPW)
- Divert Locks (PSDL)
- Use Commitment Control (PSCC)
- DBCS Capable (PSDB)
- Client to Server (PSCT) and Server to Client (PSST)

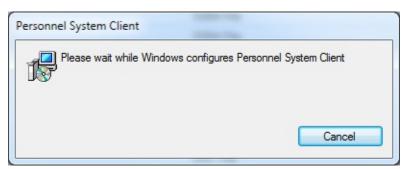
10.5 Network Client Installation

The main application installation is generally referred to as the primary installation to distinguish it from the network client installation which is a small '.msi' file automatically created for most primary installations and delivered by the primary installation. It constitutes a 'Network Client' installation and only installs shortcuts to the primary installation on a file server, from where it must be executed. It cannot be copied elsewhere and run.

To install a client application in a corporate environment, you must first install the primary application on a file server. Then provide a shortcut to the client installation '.msi' to users with access to that file server. For example:

\\SRV1\Programs\ Lansa Pty Ltd\Personnel

System\X_Win95\X_Lansa\Execute\D20A_Client_En-us.msi.



Apart from this introductory dialog and progress information, no other user interface is presented to the end user. No configuration options are required as the client uses the primary installation as the target for the shortcuts it creates. These shortcuts are defined by the primary installation's definition.

11. Deploy LANSA for the Web Applications

Considerations

- The machine where the installation is built must be a fully working example of the application. This is vital as many configuration files are pulled directly from the Build environment.
- LANSA for the Web applications on Windows consist of three logical parts: the Web Server (e.g. IIS), the Data Application server (i.e. LANSA) and the Database Server (e.g. SQL Server). It is most common, even in high transaction environments, to locate the Database Server on a dedicated machine so that as much RAM as possible may be allocated to the database. The Web Server and Data Application Server are then installed together on a separate machine, again the RAM is largely available for the LANSA processes to execute in. In systems with a high throughput, multiple Web Server/App Server machines may be installed with a Load Balancer to share the load between them, or even a simple DNS round-robin multiple IP address setup to load balance between the servers.
- Linux and IBM i servers may also host any part, or all, of LANSA for the Web. Some common scenarios which involve a part of Windows and a part elsewhere are:
- Windows Web Server with IBM i Data Application Server and DB2 i Database Server.
- Linux Web Server with Windows Data Application Server and SQL Server Database Server.
- Windows Web Server and Data Application Server with Linux Oracle Database Server
- IIS must be enabled on the Web Server. Specific settings are not required as the install automatically configures what it requires.
- The User ID for the Web Configuration must be able to log on to the Data Application Server and should be a member of the user group "LANSA Users". This is configured automatically.
- When deploying LANSA shipped weblets (typically the std_* weblets) or external resources (XW* external resources), either as selected objects or a cross-referenced objects, ensure the package has been built in an environment where the appropriate version of these objects has been installed.

- A WAM or WEBEVENT function **cannot** be automatically launched from a deployed package in a browser.
- A URL shortcut can be created to launch the Web Application using the Files with Special Processing option.
- To request a runtime License for your LANSA web application you must supply the CPU details of the Windows server on which the application is installed. This is achieved using a shipped LANSA utility, called X_CPU. Refer to the LANSA Support web page for further information.
- To review the current licenses on the runtime environment execute the process *LICENSES.
- The LANSA utility X_CPU and the process *LICENSES are included when you deploy the LANSA runtime.
- Template XWAMENB has been created to assist with the deployment of a WAM Application.
- Refer to the Installing LANSA on IBM i Guide or Installing LANSA on Windows Guide for more information on installing the Web Server.

Restrictions

- The web user is not configured to be the partition security officer Update X46SEC with the web user so that the Web Administrator may connect successfully. E.g. Set it to PCXUSER as the default.
- Default SQL Server Instance MSSQLSERVER is not fully supported If a default SQL Server instance is used, the web user will not be given permission to login to the database.
- Deploying a large set of files in the Web Images may fail with messages like "too long for an identifier"
 The images directory will need to be deployed manually.
- JSM Service must be stopped before building a version which includes Integrator
- Some shortcuts are not removed when the application is uninstalled.
- MSI Install will fail part way through if there is a pending reboot. The install will be rolled back.
 - Reboot the machine and restart the install.

Also see

11.1 Web Server Deployment

11.2 Data Application Server with Windows Application Database

11.1 Web Server Deployment

Do not use a template.

The settings required:

• Deploy LANSA communications

The options required:

- Require Elevation Yes
- Setup Database No
- Install Web Administrator
- Install IIS Plugin
- Install Web Images
- Server Name (PSLU) set to the same name used by the Build environment (in L4W3SERV.CFG).
- Communications route created of the same name as the PSLU value.
- These server names must match so that the development environment configuration options are used, otherwise a new entry is created with default settings.

Web Server objects are non-LANSA objects such as cascading stylesheets, images, and scripts used to render the browser interface. They are all shipped simply by choosing to Install Web Images.

Automatically Configured Items

• Sets up the L4w3serv.cfg file. Note that HostRouteLuName is assigned from PSLU. There is not a specific MSI Property for this. This means that it is not necessary to run the Web Administrator to setup this file.

Manual tasks after Installation

If IIS Plugin is already installed, the configuration of IIS will not be done.
 Manual steps will be required to configure the existing plugin to use the
 MSI-installed app/server or to replace the plugin with the MSI-shipped
 version.

Note that uninstalling the existing Plugin before installing the MSI may be a simpler option, and then configure the plugin to use the existing app server.

11.2 Data Application Server with Windows Application Database

Do not use a template.

Include any application objects in the package e.g. WAMs. If the Web Server is on the same machine as the Data Application Server, then also configure the Package as described in 11.1 Web Server Deployment.11.1 Web Server Deployment

The settings required:

- Deploy System Definition
- Deploy Partition Definition
- Omit Object Definitions
- Deploy Execution Database Support
- Deploy with Component Support
- Deploy with Web Environment Support
- Deploy LANSA communications
- Deploy Server Support

The options required:

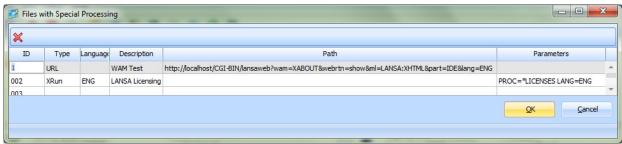
- Require Elevation Yes
- Setup Database Yes
- Install Listener
- Install Web Administrator
- Specify *LICENSES for the Process To Execute. This will remind the installer to obtain a license as well as provide the opportunity to enter the license.

The Communication Administration file (listen.dat) will need to be included to facilitate connection to the data server. This is created by selecting the Application, clicking on the Communication Administrator icon and clicking on the Listener from the menu. There is no need to create a Route, unless installing the Web Server too. Note that the Listener TCP Port will be prompted during the install.

Connection details for the database must be specified to ensure correct entries are created in the X_LANSA.PRO file.

It is not possible to launch a web application on completion of the installation; however a shortcut can be created to assist with launching the application after

it is installed. For example enter shortcut details similar to those following:



Automatically Configured Items

- The dc_w08.dat is shipped and installed from the development environment.
- Grant "Everyone" full control to the root install folder and subtree.
- Deploy with Web Environment Support ships and imports the following directories (ALWAYS whether SUDB=Y or SUDB=N):
- Lansa\Imports\std_weblets
- Lansa\Imports\WebResources
- Creates/sets up a Local User
- If SQL Server, creates that Local User as a database user and sets up appropriate access rights.

Manual tasks after Installation

- If not SQL Server, give the web user appropriate rights to logon to the database server.
- Set the web user as the partition security officer by modifying table LX_F96 column X96OWN
- The dc_w08.dat is shipped and installed from the development environment. Any modifications from the default need to be made manually after install using the Web Administrator.
 - The default values are: Any web server, from any port, with anonymous user, for any partition.

Troubleshooting

• If a connect in the Web Administrator returns error 0x1C, then Install Server Support has been omitted. A communications trace reveals failure to load LXCP0010.DLL.

12. Just In Time Upgrade Guidelines

Let's start at the beginning.

What is Just in Time Deployment?

Simplistically, Just in Time (JIT) Deployment provides a centralized, automated update facility for any application deployed using the Deployment Tool.

In Version 13 Just in Time deployment only supports Package Upgrades. DLL Upgrades are no longer supported.

All the Packages associated with an Application are installed on a centralized Application Server, that is, the initial Version and any subsequent major upgrades or Patches. After the initial installation, each time a user launches the Application from a Target PC, the Target PC connects to the Application Server to see if new upgrades are available to install. Any new upgrades will be automatically installed before the Application is launched.

How does it work?

To get started the initial Package to install the application must include:

- The application objects, the same as any other deployment
- Information defining the Application Server, so the target PC can connect to the Application Server to install future Packages

A variety of deployment templates are available to assist in the construction of these Packages.

The initial Package may be delivered in various ways including:

- file share on the network
- web site
- an ftp server
- CD or DVD

It is only after this initial installation that Just in Time Deployment really comes into play although we have now identified the Application Server and set up the Just in Time configuration to access the Application Server whenever the Application is started on the Target PC.

Subsequent updates to the application are either a major upgrade Version or a Patch.

When you are using Package Upgrades (the most common option) and you deploy a Package upgrade to the Application Server you **no longer** create an

entry for the Package in the associated APPLPKG.DAT file. It's a simple matter of just copying the MSI or MSP into the Application directory

Each time a user launches the Application on a Target PC a connection is established with the Application Server. Before the Application is started the Just in Time processing checks if there are any upgrades which have not already been applied locally. If any Packages are found they are installed before the Application is launched.

For more detailed information on the Just in Time set up refer to:

- 12.2 Package Upgrades
- 12.3 Application Servers
- 12.4 Target PCs and the Client Application
- 12.6 Just In Time Upgrade Recommended Settings
- 12.7 Troubleshooting Just in Time
- 12.8 Tracing Just in Time Connections
- 12.9 Upgrading to Version 13 using JIT Upgrade
- 12.10 Customizing Just In Time Package Upgrade

12.1 Major differences in Package 13 JIT Upgrade

- It's the same as it was in essence, though it is now just the one file to deploy, not a set of directories and files.
- Now that LANSA produces MSI and MSP files they can be deployed using Microsoft SCCM – System Centre Configuration Manager. LANSA JIT Upgrade is not required.
- SCCM is a far more capable deployment engine than LANSA JIT Upgrade, though SCCM can be more complex to set up in the first place. Thus LANSA JIT Upgrade will be kept as simple as possible leaving the SCCM functionality for more complex deployment scenarios.
- V12 Application Server can be upgraded independently to V13
- V13 Application Server will deliver V12 packages the same as the V12 Application Server.
- The MSI delivered is the latest one in the x_apps folder.
- The MSP delivered is the next one that has not been installed and matches the language of the installed MSI.
- The install may be customized by adding an optional SETUP.TXT file with parameters to add to the msiexec command line. That is, any of the properties that the MSI and/or Windows Installer exposes.
- An MSI is an independent install and an MSP is intended to be a cumulative install, so the prerequisite functionality has been removed.
- The authorized users have been removed.
- Only a single application is installed. The multiple applications options have been removed.
- JIT DLL Upgrade is no longer supported.
- Multiple languages are supported for each install
- Multilingual Text Variables are now used for all text values which appear in the MSI UI, in order to support multiple languages.

12.2 Package Upgrades

Package upgrades allow for the execution environment to be deployed and upgraded when the users starts their application.

Package upgrades allow you to keep a tight control over what is being deployed as each time you send a Package to the server to deploy you are creating a record of what has changed.

Patches are an excellent way to deliver LANSA EPCs to an Application. Only the LANSA EPC will be delivered by the Patch.

12.3 Application Servers

Any Just in Time configuration must include an Application Server.

All distributed Versions or Patches are stored on the Application Server. The Application Server then receives requests for upgrades when a user starts the application on their PC. Any new Versions or Patches (depending on your Upgrade Option UPGD) will then be sent to the client PC in response. As such, the Application Server is a simple file server, and does not require a powerful processor.

Installing an Application Server

An Application Server must have:

- A Visual LANSA installation
- A listener installed and running to receive requests from any Client PCs
- A valid LANSA SuperServer license. A single seat LANSA SuperServer license is usually sufficient as it is only used during the Just In Time Upgrade processing. After any upgrade Versions or Patches have been downloaded the license will be made available again.

₹ Tip The simplest way to set up the Application Server(s) is to install Visual LANSA.

Applying Packages to the Application Server

Each Application made available for Just in Time Package Upgrade deployment on the Application Server must include a directory structure like:

<sysdir>\x_apps\[Client App Name]\

This directory structure is a mirror of the directory structure on the deployment system.

Within the **[Client App Name]** directory **only** the MSI/MSP to be deployed is required. The original MSI used to install the Application must be included.

Warning: When the JIT Application Server is the same LANSA system as the Deployment System any distributed Package will be available to be downloaded immediately. For this reason it is recommended to use different LANSA systems for the creation and deployment of Packages to facilitate testing of the Package before it is available to download.

Multi-tier Application Servers are not supported from Version 13. Use Microsoft SCCM for complex deployments such as these.

12.4 Target PCs and the Client Application

The LANSA Application installed on each Target PC is essentially the same as any other Client Application **except** it includes a reference for the Application Server.

Connecting to an Application Server

The connection to an application server is generally accomplished using the X_RUN ASXX parameters. These parameters work identically to the PSXX parameters, as described in The PSXX= Parameter in the Technical Reference.

The SSN (Symbolic Server Name) used when automatically connecting is always *APPSERVER. This value cannot be changed. **Generally, only the ASLU and ASPW parameters need to be defined**. Refer to Just in Time Options for more information.

SuperServer applications may connect to an Application Server (via the ASXX parameters) as well as their normal server (via Built-In Functions or the PSXX parameters). Alternatively, SuperServer applications may use their normal server as an Application Server. Where more than one server connection is started via an INIT function, the first connection is used as the Application Server.

Important: If ASXX parameters are used to connect to the Application Server it is necessary to have *.XQ* files on the Application Server. Refer to The .XQ* Files in the Technical Reference for details.

Refer also to Connect to the Required Server.

Downloading Packages to the Target PC

When a Deployment Tool Package is installed on the Target PC the same directory structure mirrors the Application Server, that is the directory structure is created like:

<sysdir>\x_apps\[Client App Name]\

When the Application is launched on the Target PC:

- A connection to the Application Server is established
- The Windows Installer GUID of the current application is queried for its current state which Package and Patches are installed, and this is sent to the Application Server to determine if there is a later MSI available (only the

latest available is used), and if not, whether there is a later Patch available. Any new Version or Patch is downloaded. Any Package that previously failed to install correctly will be attempted to download and install again.

- Even if a new Version and multiple Patches are needed to be installed, only 1 is installed at a time. The application is then launched and the Application Server checked again to see if there is a Patch available. (It also checks for an MSI first, but as that has just been installed, it can only possibly find a Patch to install.)
- Patches are installed in numerical order. Every Patch is installed that matches the MSI.
- When the Client receives the MSI or MSP, it is launched using SETUP.EXE and SETUP.BAT and then x_run terminates. This always happens.
- MSI files will behave just like a manual install. That is, there will be a user interface requiring user interaction. If a silent install is required then a custom SETUP.TXT file needs to be provided on the Application Server to pass /quiet or /passive to the MSI.
- MSP files automatically run with /passive. Again you can override this by supplying a custom SETUP.TXT file.

Downloading DLLs to the Target PC

When DLLs are installed on the Target PC, the same directory structure that exists on the Deploying System and the Application Server, is created. That gives a directory structure like this:

<sysdir> $[x_ppp]$

Each DLL installed on the Target PC is downloaded to this directory and will override any existing Packages of the same DLL.

12.5 Application Server Development Life Cycle Recommendations

Sections Deployment Tool Environment and Application Servers raise the need to have a separate Application Server from the Deployment System. This is to prevent production systems being immediately updated as soon as a package has been built and before there has been time to test the deployment. But how does this work in practice? What steps are involved in producing and testing a deployment in a controlled manner?

The crux of the issue is that the package contains the connection details for the Application Server. Thus there can only be one Application Server for an application. But two separate places are needed to go to for Test and Production. How can it be varied?

For the first install only, the tester needs to specify on the MSI command line either:

- a different application server or
- a different dummy test application name.

Note that specifying an x_run parameter on the MSI command line not only changes it for the install, but it also changes the value for when the application is executed. Subsequent executions of the application will continue to use the same mechanism to access the correct packages on the correct application server.

Option (1) - a different Application Server

This option is only available from Version 13.2 onwards.

An Application Server is required on both the Deployment System and a separate Production System. Both Application Servers may be on the same machine.

The application LANSAAPP is created on the Deployment System with the Application Server name set to the Production Application Server. That is, the default behaviour is to use the Production Application Server.

Two sets of route information need to be included in the application: one for the Production Application Server and one for the Test Application Server.

The tester will then execute the first MSI using the command line. Change to the directory where the MSI is located and type <MSI filename> ASLU=TEST. For example:

LANSAAPP_1.2.3_en-us.msi ASLU=TEST

Once the package has been fully tested, it is copied to the Production Application Server.

Option (2) - a different Dummy Test Application Name

The Application Server is separate from the Deployment System. The application LANSAAPP is created on the Deployment System and copied to the Application Server but not to the $x_apps\LANSAAPP$ directory. A separate directory is created, say $x_apps\TEST$ and all the packages are put in there. The same ones as are in the production directory $x_apps\LANSAAPP$, any setup.txt file plus the new package. If there is a setup.txt file it will need to also specify APPL=TEST.

Only one set of route information is required as there is just the one Application Server.

The tester will then execute the first MSI using the command line. Change to the directory where the MSI is located and type <MSI filename> APPL=TEST. For example:

LANSAAPP1.2.3_en-us.msi APPL=TEST

Once the package has been fully tested, its copied to the Production directory on the Application Server. For example, to x_apps\LANSAAPP.

12.6 Just In Time Upgrade Recommended Settings

Upgrade

New Versions/Patches is the only functional option for **Option (UPGD)** Upgrade Option. The latest Version and all Patches not currently installed on the Client will be downloaded and installed.

> The alternate value of No Upgrade means there is no JIT processing.

Application name (APPL=)

The APPL parameter is included in the generated X_START file for the Package. The Application Name is automatically included for this argument.

Locked upgrades file)

During the install of a Package, a temporary file UPGRADE.LCK is created in the root directory of the (_upgrade.lck application being processed.

This file prevents different installations of the Application from running the upgrade simultaneously on the same PC. The file is removed when the upgrade is completed.

If a lock is found during the upgrade, the application upgrade's behavior will be governed by the settings for the *Connection failure action (UPCF=)* parameter.

Refer to the *Connection failure action (UPCF=)* in Just in Time Options for details of this parameter.

12.7 Troubleshooting Just in Time

\triangle

An error occurred while checking for application updates.



This is the most common error encountered during a Just in Time Upgrade and can be displayed for a number of reasons.

The first step to determining the cause is to examine the X_ERR.LOG file on both the JIT Application Server and the Client PC. In most cases this will point out exactly what the issue is, and corrections can be made to resolve the issue.

Refer to the following more specific issues for more information.



Cannot connect to JIT Application Server.



There are a few different areas to check in this case:

- 1. Check that the User (ASUS) and Password (ASPW) supplied are correct or Trusted Connection is selected (ASTC).
 - Ensure the User is valid as a login to the JIT Application Server.
 - Double-check the User and Password was entered correctly.
 - Check the User and Password values in the application start file on the Client PC.
- 2. Check the Host Route entry exists and is correct in the lroute.dat file using the LANSA Communications Administrator.
 - Ensure LANSA Communications has been deployed to the Client PC.
 - An appropriate entry should exist in the lroute.dat file on the Client PC. Check the LU name, Qualified Name and port number are correct.
 - Check the listener is started on the JIT Application Server and listening on correct port.
- 3. Check the network configuration
 - Verify the IP Address being used
 - Correct any network errors.



Errors encountered during the update checking.



- 1. Check the JIT Application Server has a valid SuperServer license installed
- 2. Check the directory structure
 - The directory structure in the x_apps directory on the JIT Application Server is very important and should not be modified.
 - The x_apps\<application name> folder should include the MSI or MSP files associated with the application Versions and Patches to be deployed.



New Package not downloaded as expected.



1. Use the Windows Control Panel Add or Remove Programs to determine the Application installed. Compare that with the MSI or MSP files available on the Application Server.

12.8 Tracing Just in Time Connections

If issues still exist with your Just in Time deployment after reviewing the 12.7 Troubleshooting Just in Time, the next step is to configure tracing to obtain more detailed information.

To configure tracing:

Step How to do it

1. Enable Add the following X_RUN commands to the X_LANSA.PRO

Tracing on the or X_START start file:

Client ITRO=Y Application ITRL=4

ITRM=999999

or

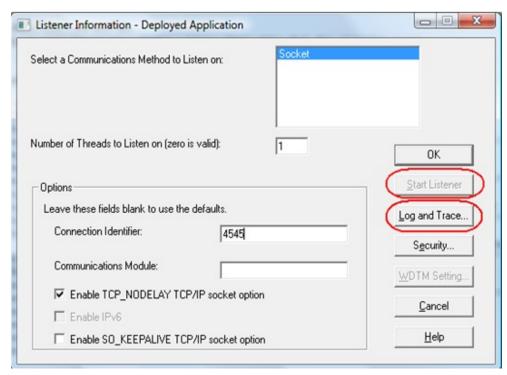
Enable tracing globally

Add the following X_RUN values to the system environment variables



2. Enable Click the Log and Trace button.

Tracing on the Listener on the Application Server



In the Trace panel, select the *Select Basic* button and then click OK.

Stop and re-start the Listener.

Close the Communications Administrator.

3. Retry the connection

If the connection continues to fails, select No when prompted to continue.

The Listener trace files can be located in the Connect directory on the JIT Application Server. Locate the log file x_lansa.trc.

The Client Application trace files should be located in the X_LANSA directory or %temp% directory as x_tracennn.txt files.

4. Contact LANSA Support

Contact LANSA support and forward your trace files and any detailed information relevant to your Just in Time setup.

5. Remove tracing

Remove all tracing to avoid overheads when running the application.

12.9 Upgrading to Version 13 using JIT Upgrade

The first step is to upgrade the Application Server to Version 13. It will continue to provide V12 Package upgrades to the V12 clients until a V13 Application is created and set up. It is critical that the Application name of the V13 Package is the same as the V12 Package. This will occur automatically if you convert your V12 Application to V13 using the feature provided in the Deployment Tool. Refer to Convert a Pre-V13 Template for more information.

When the MSI file has been placed in the correct location, the next time the V12 Client connects, the V13 server will detect that a V13 Package is available and download that. It will no longer download V12 packages.

Behind the scenes, the V13 server always supplies a SETUP.EXE to the client that executes the MSI file. This allows for customization of MSI parameters, but is crucial to the V12 Client being able to install the V13 Package. This is because the V12 Client only executes SETUP.EXE. Further, if the Client is V12, then the file x_apps\<Application name>\boot\cab2\build.dat is transferred to the Client. This causes the V12 Client to execute SETUP.EXE because the build number inside (e.g. 4051) is later than the one the Client is using (for example 4050).

12.10 Customizing Just In Time Package Upgrade

Why would you want to customise the JIT Package Upgrade?

- To provide command line parameters to the MSI to alter its behaviour, like a silent install, or logging parameters.
- To have different users on different upgrade paths. Say, some users receive a Patch but others don't.
- To provide your own SETUP.EXE to do whatever you want.

A SETUP.EXE is shipped with LANSA 13.0 in the execute directory. This program runs a file called SETUP.BAT from the same directory as SETUP.EXE is executed from.

An example of the automatically generated SETUP.BAT for a patch is:

```
"TESTMSI2_v2.3.4.2_en-us.msp" /passive
if errorleve 1 exit start ""
"C:\TestMSI2\X Win95\X Lansa\Execute\X Run.exe"
UPCD=815F0A60-A9EC-41AB-A755-858C28F00C2B
LANG=ENG PROC= FUNC= FORM=VL DEM20
PART=DEX USER=QPGMR INDB=N INST=MSI
UPSI=YES ASLU=*LOCAL ASUS= ASPW=
ASTY=*OTHER ASCT= ASST= UPCF=PROMPT
UPDF=PROMPT ASKC=NO ASTC=YES
DBUT=MSSQLS DBII=TESTMSI2 DBUS= PSWD=
INIT= PSTY= PSLU= PSUS=QPGMR PSPW= PSEA=
PSCC= PSDB= PSDL= PSWM= PSEP= PSCT= PSST=
PSTC=NO APPL=TESTMSI2 DBID= PSNM= UPGD=Y
UPTP=P CMTH=T XENV=X DOLLAR SIGN CHAR=$
XENV=X HASH SIGN CHAR=#
XENV=X_AT_SIGN_CHAR=@
XENV=X_GEN_AT_SIGN_CHAR=@
```

The first line is the MSI or MSP to be run - in this case:

"TESTMSI2 v2.3.4.2 en-us.msp"/passive.

The Application Server generates this line. The second line contains the parameters of the client that is running JIT. This line is added by the **Client** before calling SETUP.EXE. This launches the application after the installation of the Patch completes. SETUP.EXE uses the API ShellExecute to execute SETUP.BAT using the verb 'open'.

Patches use /passive because they don't have any questions to answer, so why require the user to enter anything? The /passive option displays a progress bar so the user gets feedback that something is happening.

When **setup.txt** has /passive specified, an MSI install will NOT start x_run after installing.

/passive prevents the Exit Dialog being used and thus x_run is not started. The end user will need to restart the application manually.

So that's the automatic operation.

If the file "**SETUP.TXT**" exists in the Application directory, the contents are appended to the MSI/MSP filename, for example:

/L*V+ "C:\package.log"

The parameters specified are MSI Public Properties. Please refer to msiexec.exe Command-Line Options for details.

Any valid MSI/MSP parameters can be used here. This particular example logs everything, appending to the file c:\package.log. This is the first level of customisation - add parameters to the MSI command line. The previous SETUP.BAT would therefore become:

```
"TESTMSI2_v2.3.4.2_en-us.msp" /L*V+
"C:\package.log"
if errorlevel 1 exit
start ""
"C:\TestMSI2\X_Win95\X_Lansa\Execute\X_Run.exe"
UPCD=815F0A60-A9EC-41AB-A755-858C28F00C2B
LANG=ENG PROC= FUNC= FORM=VL_DEM20
PART=DEX USER=QPGMR INDB=N INST=MSI
UPSI=YES ASLU=*LOCAL ASUS= ASPW=
ASTY=*OTHER ASCT= ASST= UPCF=PROMPT
UPDF=PROMPT ASKC=NO ASTC=YES
DBUT=MSSQLS DBII=TESTMSI2 DBUS= PSWD=
INIT= PSTY= PSLU= PSUS=QPGMR PSPW= PSEA=
PSCC= PSDB= PSDL= PSWM= PSEP= PSCT= PSST=
PSTC=NO APPL=TESTMSI2 DBID= PSNM= UPGD=Y
```

UPTP=P CMTH=T XENV=X_DOLLAR_SIGN_CHAR=\$
XENV=X_HASH_SIGN_CHAR=#
XENV=X_AT_SIGN_CHAR=@
XENV=X_GEN_AT_SIGN_CHAR=@

The JIT Application Server checks if there is an x_apps directory for the requested Application. If there is, then an MSI will be downloaded. If not, the x_pkgs directory is used. This allows for the support of pre-13 clients by a 13.0 server.

The server then checks if SETUP.EXE is in the Application directory. If it isn't then the copy in the LANSA execute directory is transferred to the Client. This is another level of customisation available. A developer can put whatever SETUP.EXE file they like in the Application Directory and it will be transferred to the client and executed. For example it could be a self-extracting executable as a way to get other files downloaded to the client.

Of course, your custom SETUP.EXE will need to run SETUP.BAT if it is also going to install the MSI/MSP and then launch the application. Support by LANSA of this configuration only extends to delivering the custom SETUP.EXE to the Client. What it does after that you will need to support yourselves.

So, the files transferred to the Client are:

- SETUP.EXE
- SETUP.BAT
- <The next MSI/MSP file that the client does not have installed>

Further, if the Client is pre-13.0, then the file **x_apps****Application name**>**boot\cab2\build.dat** is created in the server's LANSA temporary directory (usually %TEMP%) and this is transferred to the Client. This causes the pre 13.0 Client to execute **SETUP.EXE** because the build number inside (e.g. 4051) is later than the one the Client is using (e.g. 4050).

The Application Server can have an Application Directory that does not match the original Application name. The Client needs to install the first MSI specifying the new name as an MSI property. This will set the APPL value to the new one and attach to JIT using the new APPL. The Server will generate a SETUP.BAT that specifies APPL on the MSI command line.

APPL is a runtime only value that can ONLY be set on the MSI command line. It is only necessary when you have a JIT server. The 1st install of the application

needs to set the value too. That is done by supplying the 3 files (SETUP.EXE, SETUP.TXT and the MSI) that the JIT provides to the Client and just getting the Client to run SETUP.EXE rather than the MSI. This is the recommended approach to ensure consistency of APPL value. An alternative is to provide instructions for the Client to run the MSI using msiexec.exe and the property value to pass. Or a batch file can be provided.

The server is set up by creating the new APPL directory name. Copy the MSI file from the original directory. JIT will now correctly not find any updates to install on the client. Copy MSP files in as required and JIT will install them the next time the application is executed. There must be at least one MSI or MSP in the directory otherwise a 750 error is output to X_ERR.LOG on the server (0750 - Application testmsi3 not found or no packages in applpkg.dat)

In summary, the customisation available is:

- 1. Create SETUP.TXT yourself and specify the MSI parameters, including all the X_RUN parameters exposed as MSI Properties.
- 2. The SETUP.EXE can be replaced and perform whatever install the developer desires
- 3. Use a different Application Name than the one it was built with for multiclient support at different Package levels. For example, Application built as LANSAInventoryControl could be put in directories

 Customer1InventoryControl and Customer2InventoryControl. These can then be maintained at different levels even though they are running exactly the same software. They can also be given different Patches too.

Example setup.txt files

1. /**qb**

Displays a progress dialog only for both MSI and MSP installs and no confirmation dialog. Thus the installs happen automatically without any user interaction. MSI installs will use the settings from the previous install.

2. /l*vx %TEMP%\InstallLog.txt

Diagnostic log file output to user's temporary directory.

3. /qf

Displays a full user interface which overrides the default Patch behaviour of /passive and also overrides the Silent Install setting inside the MSI/MSP.

13. Deployment Templates

Deployment Templates allow you to define default settings and their appearance and save them to be used as the basis for a new Package. Almost every subset of a Package's options and settings can be defined in a template. However, certain areas, such as Package dependencies, cannot be retained in a template as they pertain to specific application or Package information.

Why use deployment templates?

- If you are deploying using the same deployment scenario repeatedly, saving the initial Package as a template and then using this template to create subsequent Packages saves time and ensures consistency in the deployment options and settings.
- By using templates, you can greatly limit the amount of information displayed on the Package Control Panel - thus reducing the amount of time it takes to define or assemble a new Package. You can specify that an option is hidden from view, not allowed to be changed, or be visible and editable. This means that there may only be a handful of options to select and some of these options may be highlighted as "must be considered".

A number of predefined templates are supplied with the Deployment Tool. These are described in 13.5 Supplied Deployment Templates.

Also see

- 13.1 Create a Deployment Template or 13.3 Convert a Pre-V13 Template
- 13.2 Review, Change or Delete a Deployment Template
- 13.4 Deployment Template Options & Settings

13.1 Create a Deployment Template

There are three different ways to create a deployment template.

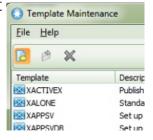
Step How to do it

1a. Selecting the Templates menu on the Main Window.

Manually define a

From the Template Maintenance request a new template **b** on the toolbar or from the File menu.

Deployment Template

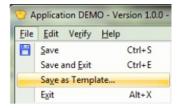


OR

1b. Copy a When you have created a Version or Patch to suit your Package as deployment scenario, you can use this as the basis for a

deployment template.

Deployment To do this, with the Version or Patch open in the *Control Panel*, Template select the *Save as Template* command from the File menu.



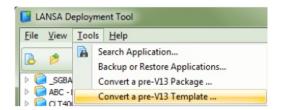
OR

1c. Convert If you created your own template in an earlier version of LANSA a pre-V13 these can be converted to V13 Templates.

To do this select the Convert a per-V13 Template command

To do this, select the Convert a per-V13 Template command from the main Deployment Tool window.

Important – Do NOT convert any of the LANSA supplied templates as these have been redesigned in Version13 and many of the old templates are no longer relevant.



2. Create the Deployment Template

The New Template dialog will be opened to define the basic template properties.



Template Name

Enter a name for the Application in eight characters or less. The first character must be an alphanumeric character and the name cannot contain embedded blanks.

Template Description

Enter a description that will enable you to easily identify this template in the list of templates.

Press *OK* when complete. The *Template Control Panel* will be opened as shown in 13.4 Deployment Template Options & Settings.

13.2 Review, Change or Delete a Deployment Template

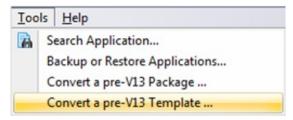
To review, change or delete an existing Deployment Template, open the Template Maintenance dialog from the Deployment Tool's main window. Select the template you want to work with in the list of templates then either Open the template \red to review or modify the definition or delete the template \red as required.

An alternate way to open a template is to simply double-clicking the template in the list of templates.

A Package based on a template retains an association with the template after it is created. This may restrict the options available for selection and modification. Refer to Template associated with Package for more information.

13.3 Convert a Pre-V13 Template

If you developed templates in a pre-V13 LANSA environment, these templates can be converted to be usable in Version 13. In the main window, select the Convert a pre-V13 Template option from the Tools menu to locate and convert a template defined in an earlier version of LANSA.



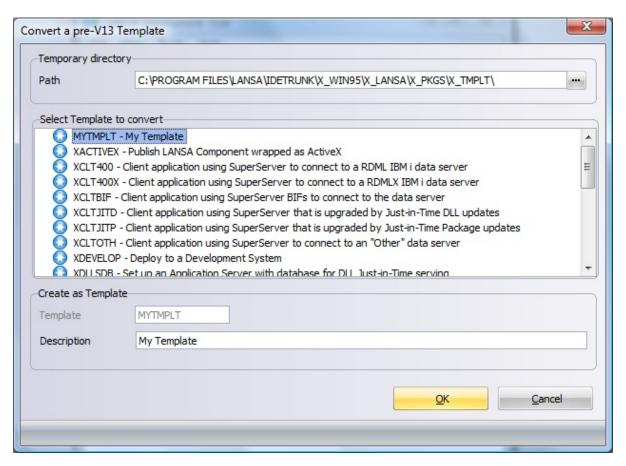
Before converting any template ensure you have understand The Deployment Model and the changes in the deployment paradigm in Version 13.

Important – Do NOT convert any of the LANSA supplied templates as these have been redesigned in Version13 and many of the old templates are no longer relevant.

The conversion interface is a three step process:

- 1. Enter path where template to be converted exists
- 2. Select a template to convert.
- 3. Confirm and modify details as required.

When details are complete select OK to convert the selected Template.



Template Template is pre-filled from the copied Template and cannot be modified.

Description Enter a Template description to detail what this template implements.

13.4 Deployment Template Options & Settings

When you create a template or open an existing template you will see a window that looks very similar to the Control Panel. However some options, like Files with Special Processing, are not included in the template definition as they are Package specific information which cannot be generalized in any meaningful way.

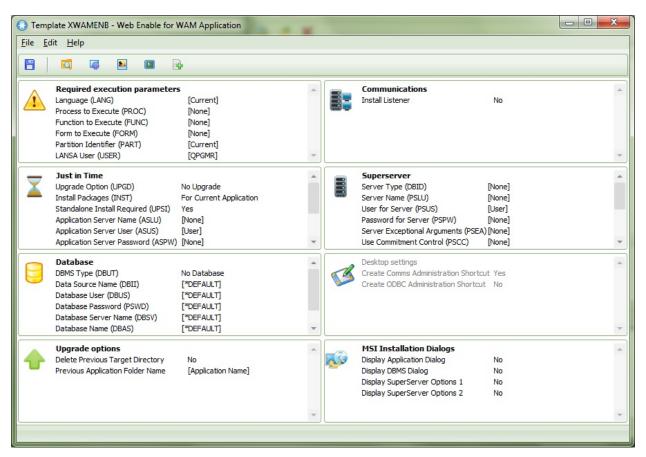
If you have created a template by copying an existing Version or Patch the 13.4.1 Template Installation Options displayed will reflect the base settings and options. You may choose to review the template and hide or highlight specific options.

To understand how values and appearance are controlled by templates refer to 13.4.2 Template Option Values and Appearance.

13.4.1 Template Installation Options

• Click on an area in the screen picture to jump to the details.

Select the Installation subsets from the main area of the window and the settings from the toolbar to specify the options you wish to use for your template. For further details about these options please refer to the description of the equivalent option in the Control Panel.



The options on this window have been grouped so that, for the majority of groups, a whole group can be ignored and default values assumed if they are not relevant to your Application install. For example, if the Application is not using the *Just in Time* facilities, the defaults can be accepted, and you can then ignore the whole *Just In Time* group. Similarly, if you are not deploying to a local database, the *Database* options can be ignored.

13.4.2 Template Option Values and Appearance

Each option in the Control Panel can be configured to have a specific default value and to indicate how or even if the option is displayed.

Similarly each subset of options in the Control Panel can be configured to indicate how, or even if, the whole subset is displayed. Refer to 13.4.3 Template Subset Appearance for more information.

The default values for installation options are enclosed in square brackets; for example, Language (LANG=) has a default value of [Current]. **DO NOT enclose your installation option values in square brackets**. If you do, they will be ignored and Deployment Tool's default value will be used for that option.

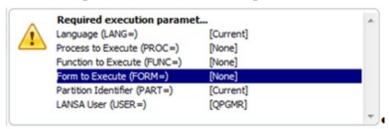
Step How to do it

1. Modify Template Installation Option(s)

Create or open a template.

Double click a Template Installation Option to review the definition.

For example, double click the option Form to Execute:

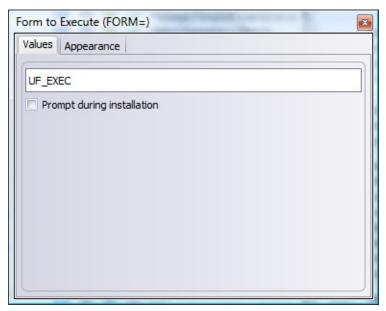


Refer to 13.4.1 Template Installation Options for information on specific options.

2. Set up the Values default

The *Values* tab allows you to specify a default value to be applied for an option when the template is used.

If the option is a free-form entry, like Form to execute (FORM=), an edit box will be displayed. Change the value to the default value you want to be applied each time the template is used to create a Version.



In this case you can also decide if you want this value to be prompted, thus enabling it to be changed during the installation process. Where relevant, some options also include Prompt during X_START for selection.

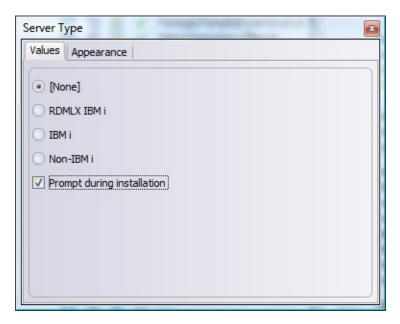
Prompt during Installation

When installing the Package using Windows Installer you will be prompted to enter or confirm the value for this option.

Prompt during X_START

When running the application using an X_RUN (typically from a desktop icon) you will be prompted for X_START values when the application is launched. The values entered will be used as default values on the X_START prompt.

When the option has a predefined list of values, a series of radio button is displayed in the values tab.



3. Review the Appearance default

The *Appearance* tab allows you to control how the option or subset of options will be displayed when the template is used.

Appearance Option will be visible and editable

The option will behave as normal and will be selectable and editable.

Hide option

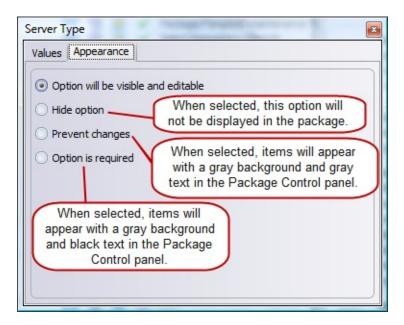
The toolbar icon for the option will be disabled. Any value associated with the option in the Template will be applied to the Package but access to these values will be denied.

Prevent changes

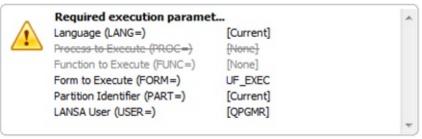
The option can be seen but cannot be changed. Any corresponding Toolbar icon will be available on the *Package/Template Control Panel* toolbar, but the *OK* button in the dialog will be disabled. Any changes attempted will be ignored.

Option is required

The required option is displayed with a gray background to indicate that it should be considered before the Package is built.



4. Save Values and Appearance settings To save the Values and Appearance settings simply close the dialog. The template interface will indicate the appearance applied to each option.



5. Set up Template Subset Appearance Refer to 13.4.3 Template Subset Appearance for more information.

6. Set up other Package settings and options

Set up any other default information required in the template by reviewing the various options and settings available in the toolbar.

For example, the template may be set to use your Company's Icon as the default Application Icon.

Refer to Package Options and Settings for more information.

7. Save the Save the template definition using the Save option on the Deployment toolbar or File menu or by using the shortcut key Ctrl + S.

Template You can also Save and Exit in one step from the File menu or

using the shortcut key Ctrl + E.

To Exit without saving use the Exit option from the File menu, the shortcut key Alt + X or just close the Template Control Panel dialog.

13.4.3 Template Subset Appearance

Each subset of options in the Control Panel can be configured to indicate how, or even if, the subset is displayed when the template is used to generate a Version. This is a very useful way to remove any subsets of information from the screen if they are not relevant for the template implementation.

There are eight subsets of options on the Template Control Panel:

- Required Execution Parameters
- Communications
- Just in Time
- SuperServer
- Database
- Desktop Settings
- Upgrade Options
- Installation Dialogs

To configure the appearance of a template subset:

Step How to do it

1. Modify Template Subsets

Open or create a template. Double click a Template Subset to review the definition.

For example, you could configure the whole subset Required Execution Parameters:



2. Review the Appearance default for the Subset

The *Appearance* tab allows you to control how, and even if, the subset of options will be displayed when the template is used.

Option will be visible and editable

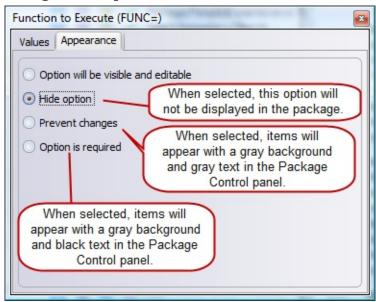
The subset of options will behave as normal. All options, unless modified at an individual level, will be selectable.

Hide option

The subset of options will not be displayed when the template is used to create a Version. Any values for specific options in this subset will be applied to the Package but will not be available to be reviewed or changed.

Prevent changes

Options can be seen but cannot be changed. The corresponding subset will be visible in the *Package Control Panel*, but the associated options will be disabled. Any changes attempted will be lost.



4. Save Appearance settings

To save the Appearance settings simply close the dialog and save the template.

5. Modify other template settings and options

Refer to 13.4.2 Template Option Values and Appearance.

13.5 Supplied Deployment Templates

The Deployment Tool is supplied with a number of pre-defined Templates to help you create Packages efficiently and accurately. If you modify any of these templates it is recommended to save them with a different name to avoid your changes being overridden during future upgrades of LANSA.

The supplied templates are:

- 13.5.1 XACTIVEX Publish LANSA Component wrapped as ActiveX
- 13.5.2 XALONE Standalone system
- 13.5.3 XAPPSV Set up an Application Server without database for Just In Time Package serving
- 13.5.4 XAPPSVDB Set up an Application Server with database for Just In Time Package serving
- 13.5.5 XCLT400 Client application using SuperServer to connect to an IBM i data server
- 13.5.6 XCLT400X Client application using SuperServer to connect to a RDMLX IBM i data server
- 13.5.7 XCLTBIF Client application using SuperServer Built-In Functions to connect to the data server
- 13.5.8 XCLTJIT LANSA SuperServer application updated by Just in Time
- 13.5.9 XCLTOTH SuperServer application connected to an "Other" data server
- 13.5.10 XEXPORT Export internal definitions
- 13.5.11 XOTHOBJ Deploy Non-LANSA Objects
- 13.5.12 XSRVNET Set up a Network server
- 13.5.13 XSRVOTH Set up an "Other" data server
- 13.5.14 XTRNAPP Translation Application with Object Details for Translation
- 13.5.15 XTRNEXP Export Object Details for Translation
- 13.5.16 XWAMENB Web Enable for WAM Application

13.5.1 XACTIVEX - Publish LANSA Component wrapped as ActiveX

This template assists with the deployment of Visual LANSA components to be exposed as ActiveX controls on Microsoft Windows. It is assumed the target PC does not have Visual LANSA installed.

Several of the option groups have been hidden as this template is designed solely to publish an ActiveX Component.

The crux of this template is contained in the Commands to Execute After Installation and selection of option Deploy with Component Support.

The after Package installation command processing

- Register the LANSA Component for ActiveX usage on the target system
- Automatically create the session configuration file required for it to be executed.

⚠ The after commands must be amended to correctly identify the ActiveX DLL and modify the configuration file settings as required before un-commenting the associated commands.

The Package settings include:

- Deploy with Component Support
- Deploy with Server Support (or just include x_dll.dll as a Non-LANSA object)

If the ActiveX functionality requires database access you should also consider settings:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database Support.

Deploy with Component Support includes X_ActiveX.dll and X_RunDllServer.dll however these DLLs must be registered in order for LANSA ActiveX to be used. This step is detailed in the After Install Commands in the template. If installing under UAC an Administrative install is required to register the DLLs.

Other settings must be configured as required.

Refer to Deploy LANSA Components published for ActiveX for more information.

13.5.2 XALONE - Standalone system

Use this template to deploy an application intended to run as a standalone Windows application with its own database. Create one application containing all the objects required to execute the application.

The Package settings include:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database Support
- Deploy with Component Support

In addition:

• DBMS Type (DBUT =) is set as MSSQLS. Modify this setting as required.

13.5.3 XAPPSV - Set up an Application Server without database for Just In Time Package serving

An alternative approach for setting up an Application Server is to install LANSA on the Just-in-Time Application Server. This will automatically set up the listener which is the basic element required on the Application Server in this scenario.

This template has the basic settings that are required to create a Version that will be used to set up an Application Server with no database for Just In Time Package serving.

This template will create a simple listener Package to be deployed to an Application Server. It is not a Just in Time Package but is used to set up the JIT Application Server.

• It is not intended to include any objects but must be deployed with the Communications Administrator to install correctly and facilitate connection to the data server. A desktop icon is created to access the LANSA Communications Administrator.

Install Listener is set as Yes.

Process to Execute (PROC) is set as *licenses to launch the Server Licenses interface as a reminder to request and apply appropriate licenses to the JIT server.

X_RUN Argument RNDR=X to ensure Server Licenses interface is executed as DirectX application.

The Package settings include:

- Deploy to Client without database
- Deploy with Component Support
- Deploy LANSA Communications
- Deploy Server Support

This template replaces XDLLSRV.

Refer to Application Servers and Deploy LANSA Communications for more information.

13.5.4 XAPPSVDB - Set up an Application Server with database for Just In Time Package serving

An alternative approach for setting up an Application Server is to install LANSA on the Just-in-Time Application Server. This will automatically set up the database and listener which are the basic elements required on an Application Server.

This template has the basic settings required to **create a Version** to **set up** an Application Server **with a database** for Just In Time Package serving. It is an extension of the XAPPSV template.

This Package is not essentially a Just in Time Package but is used to set up the JIT Application Server.

Install Listener is set as Yes.

Process to Execute (PROC) is set as *licenses to launch the Server Licenses interface as a reminder to request and apply appropriate licenses to the JIT server.

X_RUN Argument RNDR=X to ensure Server Licenses interface is executed as DirectX application.

The Package settings include:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database Support
- Deploy with Component Support
- Deploy LANSA Communications
- Deploy Server Support

The database options include:

• Database Type of MSSQLS.

The user is prompted to select and confirmed the database details.

Typically the Communication Administration file (lroute.dat) will be included with this Package to facilitate connection to the data server.

This template replaces XDLLSDB.

Refer to Application Servers and Deploy LANSA Communications for more information.

Application servers do not require a database to be installed UNLESS you intend to run the application on the Application server itself.

13.5.5 XCLT400 - Client application using SuperServer to connect to an IBM i data server

The recommended template is XCLTBIF and use BIFs to connect to the data server.

This template provides the basic settings required to create a Version for an application that will be connecting to an IBM i data server using LANSA SuperServer. The template is designed for an application with no local database.

The Server Type is set as IBM i and cannot be changed.

The Package settings include:

- Deploy to Client without local database
- Deploy System definition
- Deploy Partition definitions
- Omit Object definitions
- Deploy with Component Support
- Deploy LANSA Communications

Nominate a Server name to connect to and ensure this exists in the Communication Administration file. Typically the Communication Administration file (lroute.dat) will be included with this Package to facilitate connection to the data server.

Ensure the Client to Server (PSCT) and Server to Client (PSST) tables are set up correctly.

For further guidance refer to SuperServer Application connecting to an IBM i

13.5.6 XCLT400X - Client application using SuperServer to connect to a RDMLX IBM i data server

♀ The recommended template is XCLTBIF and use BIFs to connect to the data server.

This template is a variation of the XCLT400 template with the Server Type set as RDMLX IBM i.

13.5.7 XCLTBIF - Client application using SuperServer Built-In Functions to connect to the data server

This template is designed to deploy a LANSA application that will connect to a remote data server using the SuperServer Built-In Functions.

The SuperServer options group has been hidden as this template is designed to use the SuperServer Built-In Functions in the LANSA application to connect to the server. Some other option groups have also been hidden.

The Package settings include:

- Deploy to Client without local database
- Deploy System definition
- Deploy Partition definitions
- Omit Object Definitions
- Deploy with Component Support
- Deploy LANSA Communications

Typically the Communication Administration file (lroute.dat) will be included with this Package to facilitate connection to the data server.

13.5.8 XCLTJIT - LANSA SuperServer application updated by Just in Time

The recommended template is XCLTBIF and use BIFs to connect to the data server.

This template has the basic settings for a Just in Time upgrade to apply Package updates to the application client.

The Package settings include:

- Deploy with Component Support
- Deploy LANSA Communications

Just in Time Options include:

- Upgrade Option (UPGD) of New Versions/Patches
- Application Server Type of Non-IBM i

The initial Version is typically installed on the application server then each target PC would install the Application from this location. Subsequent Version or Patches will be automatically installed on the target PC from the Application server when the Application is launched.

13.5.9 XCLTOTH - SuperServer application connected to an "Other" data server

♀ The recommended template is XCLTBIF and use BIFs to connect to the data server.

This template is a variation of the XCLT400 template with the Server Type set as non-IBM i and Server Exception Arguments (PSEA) are highlighted and should be provided as required.

Refer to PSxx Server Parameters for more information.

13.5.10 XEXPORT – Export internal definitions

This template is designed to export the internal object definitions (fields, components, files etc...) to be imported into another LANSA repository. This template will not generate an MSI file.

All the install options have been hidden as Packages cannot be installed.

The Package settings include:

- Deploy to Development System
- Omit the Application compiled objects

13.5.11 XOTHOBJ - Deploy Non-LANSA Objects

This template has the basic settings that are required to create a Version to deploy non-LANSA objects to a specified location. This template may be useful for setting up a Web Server with required objects such as images, cascading stylesheets and scripts if these files are not defined as External Resources.

The Package settings include:

• Deploy with Component Support

These Package settings are the minimum requirements to install the Version. Most Package options are not relevant for installing non-LANSA objects.

13.5.12 XSRVNET - Set up a Network server

This template has the basic settings that are required to create a Version for installing an Application on a Network Server.

The Package settings include:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database support
- Deploy Execution add-ons
- Deploy with Component Support

In addition:

- DBMS Type (DBUT =) is set as MSSQLS. Modify this setting as required.
- Install Listener is set as YES
- Display DBMS Dialog is set as YES to allow set up of database

This template replaces XNETSRV.

13.5.13 XSRVOTH - Set up an "Other" data server

The simplest way to set up a non-IBM i Data Server is to install Visual LANSA.

This template has the basic settings that are required to create a Version for an Application that will be used to set up a Non-IBM i data server. This type of server would have SuperServer clients.

The Package settings include:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database support
- Deploy Execution add-ons
- Deploy with Component Support
- Deploy LANSA Communications
- Deploy Server Support
- Install Listener is set as YES
- Display DBMS Dialog is set as YES to allow set up of database

In addition:

- DBMS Type (DBUT =) is set as MSSQLS. Modify this setting as required.
- Review other database settings which may be required.

This template replaces XSSVSRV.

13.5.14 XTRNAPP – Translation Application with Object Details for Translation

Use this template to deploy the LANSA supplied Translation Tool as a standalone application. Include repository objects requiring translation then distribute to translators.

This template is designed to produce a standalone alone application for the LANSA Translation Tool. There is no requirement for a database.

The Package settings include:

- Deploy to Client without local database
- Omit Object Definitions
- Deploy with Component Support
- Deploy with Translation List
- Omit the Application Compiled Objects

Other options values:

- Form to execute is set to XPFTRLT (the LANSA supplied DirectX form to launch the application)
- DBMS Type is set as no database
- X_RUN value RNDR=X supplied to execute application as DirectX.

Modify the selected Languages to include the base language and any languages requiring translation.

Refer to Translating a LANSA-developed Application for more information.

13.5.15 XTRNEXP – Export Object Details for Translation

Use this template to export the details of objects to be translated. Include repository objects requiring translation then distribute the generated text file to translators.

This template is designed to produce a standalone alone application for the LANSA Translation Tool. There is no requirement for a database or any package options to be selected.

The Package settings include:

- Deploy to Development System
- Omit Object Definitions
- Deploy with Translation List
- Omit the Application Compiled Objects

Modify the selected Languages to include the base language and any languages requiring translation.

Refer to Translating a LANSA-developed Application for more information.

13.5.16 XWAMENB – Web Enable for WAM Application

Use this template to web enable the application runtime environment to support a WAM based application.

The Package settings include:

- Deploy System definition
- Deploy Partition definitions
- Deploy Execution Database Support
- Deploy with Component Support
- Deploy with Web Environment Support
- Deploy LANSA Communications
- Deploy Server Support

In addition:

- Install Listener is set as Yes.
- Install Web Administrator is Yes.
- Install IIS Plugin is Yes.
- Install Web Images is Yes.
- Process to Execute (PROC) is set as *licenses to launch the Server Licenses interface as a reminder to request and apply appropriate licenses to the JIT server.
- X_RUN Argument RNDR=X to ensure Server Licenses interface is executed as DirectX application.
- Setup Database (SUDB)is Yes.
- DBMS Type (DBUT) is set as MS SQL Server. Modify this and related database settings as required to ensure internal definitions are imported.

14. Translate a LANSA-developed Application

Typically LANSA application development, or any type of application development for that matter, is performed in a single language. For multilingual applications the task of translating the application into the appropriate languages must be completed and all translations incorporated into the application before it is distributed to users.

To facilitate the translation process the LANSA Application Deployment Tool provides a simple mechanism to package the LANSA objects requiring translations. The package is then distributed to the appropriate translators with a simple translation interface provided to simplify the process.

Further information

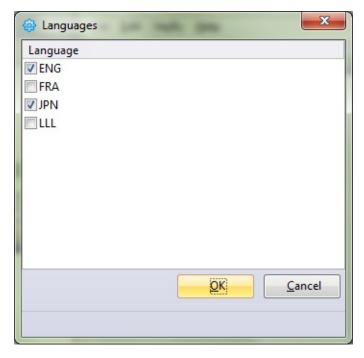
- 14.1 Translate using standalone Translation Application
- 14.2 Translate with Visual LANSA development environment
- 14.3 Load Translations for Appropriate Language

14.1 Translate using standalone Translation Application

- 14.1.1 Deploy Translation Application including Objects to Translate
- 14.1.2 Install Translation Application and Translate Objects

14.1.1 Deploy Translation Application including Objects to Translate

Step	How to do it
1. Create an application	Create an Application to contain the objects to be translated.
2. Create a Version (or Patch)	Create a Version using the template XTRNAPP if translator requires a standalone translation application.
	This template sets the <i>Deploy with Translation List</i> in the Settings, which indicates the LANSA internal definitions for the selected Objects are supplied with the Package.
	This template is designed to install the resulting package as a standalone application. Refer to XTRNAPP – Translation Application with Objects details for Translation for more information.
	A Patch can be created based on an existing Version created with this template. Note – if translator already has the Translation Application you may choose to provide only the objects requiring translation. Refer to 14.2.1 Deploy Objects to Translate for more information.
3. Add objects for translation	Use the <i>Select Repository Objects</i> adialog to add LANSA objects requiring translation to the Package.
4. Add languages to be translated	Select the Languages to be translated. Include the base language to be used as source for translation and any language requiring translations.



- 5. Save Save ☐ the Package. package
- 6. Build the Version or Patch. package
- 7. Distribute the Version or Patch ✓ to your translator(s) by Distribute providing the generated MSI (or MSP) file.

 Application to

 Translators

The files are translated as described in 14.1.2 Install Translation Application and Translate Objects.

14.1.2 Install Translation Application and Translate Objects

Step How to do it

1. Install the translation application

Run the provided MSI (or MSP) file to install the translation application on the translator's system.

- The install will automatically launch the application on completion.
- The application can be launched from the desktop.
- 2. Load the object translation information

When the application is executed the Language selection dialog is automatically launched. Select to load translation from, by default,

 $\label{lem:company} \begin{tabular}{l} \begin{ta$

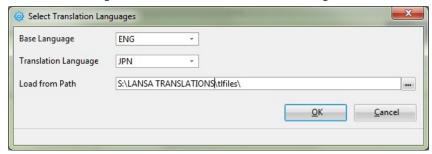
or nominate a location where the translation files are available.

If this utility has not been used before the default location for translation files will be the partition source directory associated with the installed Package.

The translation languages available for selection is based on the languages selected when the package was created.

Note - If the tl*.txt translation files are not located at the nominated location an error is issued and no languages are available for selection. Change the Load from path to the location of the translation files or move the translation files to the specified location and attempt to open files again. The translation files must include the file TLLANGS.txt.

Press *OK* to proceed with the translation process.



Base language

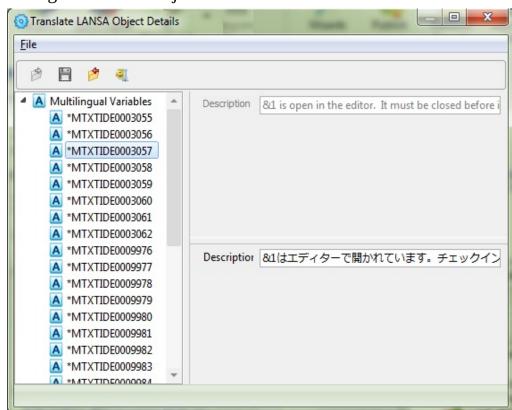
The Base language is normally the language used when developing the application. The language entered is used to locate the file on which you will base your translations. The values from this file are displayed in the upper half of the *Translate Object Details* dialog.

Translation Language

Select the language for the new translations. The languages available for translation are based on the source LANSA system and are provided in translation file TLLANGS.txt.

3. Translate If you have included LANSA objects for translation, simply work Repository through the list of Objects in the left of the window.

Object details



For each selected Object, the text in the base language will be shown in the top section of the window and the entries requiring translation are shown in the bottom section.

If the translated description is longer than the base language description the translation will be highlighted in red.

Save 🗎 your translations periodically as a safeguard.

4. Save After you have translated all the objects *Save* 🗎 the translations

your and *Close* * the files.

translations If you attempt to Close the file without Saving first your

translations will be lost. You will be warned if there are any

unsaved changes and given the option to save them.

5. Zip the *Optionally zip* the tl*.txt translation files.

translations

6. Return The generated zip file TLFILES.zip or unzipped tl*.txt

translation translation files should be returned for incorporation into the

files development environment.

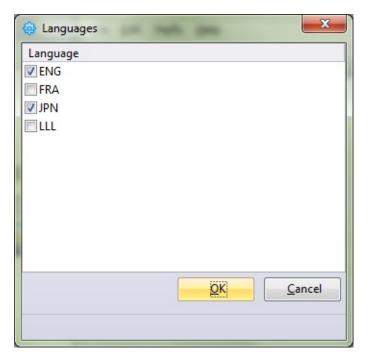
Integration of the files back into the development environment is described in 14.3 Load Translations for Appropriate Language.

14.2 Translate with Visual LANSA development environment

- 14.2.1 Deploy Objects to Translate
- 14.2.2 Run the Translation Application and Translate Objects

14.2.1 Deploy Objects to Translate

Step	How to do it
1. Create an application	Create an Application to contain the objects to be translated.
2. Create a Version (or	Create a Version using the template XTRNEXP if translator has access to Visual LANSA environment
Patch)	This template sets the <i>Deploy with Translation List</i> in the Settings, which indicates the LANSA internal definitions for the selected Objects are supplied with the Package.
	This template is designed to install the resulting Package into an existing LANSA system. Refer to XTRNEXP – Export Object Details for Translation for more information.
	A Patch can be created based on an existing Version created with this template.
3. Add objects for translation	Use the <i>Select Repository Objects</i> adialog to add LANSA objects requiring translation to the Package.
4. Add languages to be translated	Select the Languages to be translated. Include the base language to be used as source for translation and any language requiring translations.



- 5. Save package Save 🖹 the Package.
- 6. Build package Build the Version or Patch.
- 7. Distribute the Distribute the Version or Patch ✓ to your translator(s) by objects to be providing the files generated in <sysdir>\X_APPS\ translated to Checkbook Translators Checkbook Translator Translator

The files are translated as described in 14.1.2 Install Translation Application and Translate Objects.

14.2.2 Run the Translation Application and Translate Objects

Step How to do it

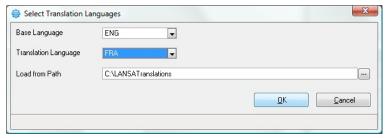
- 1. Launch Translation Application
- Log on to Visual LANSA. Ensure you have authority to edit objects to be imported or log in as security officer.
 - Select *Translate* Translate from the *Tools* ribbon.
- object translation

2. Load the When the interface is launched the Language selection dialog is automatically launched and will attempt to load the translation files from the last location used. If this utility has not been used information before the default location for translation files will be the partition source directory.

> The translation languages available for selection is based on the languages selected when the package was created.

Note - If the tl*.txt translation files are not located at the nominated location an error is issued and no languages are available for selection. Change the Load from path to the location of the translation files or move the translation files to the specified location and attempt to open files again. The translation files must include the file TLLANGS.txt.

Press *OK* to proceed with the translation process.



Base language

The Base language is normally the language used when developing the application. The language entered is used to locate the file on which you will base your translations. The values from this file are displayed in the upper half of the *Translate Object* Details dialog.

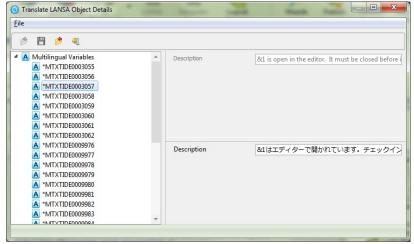
Translation Language

Select the language for the new translations. The languages

available for translation are based on the source LANSA system and are provided in translation file TLLANGS.txt.

3. Translate If you have included LANSA objects for translation, simply Repository work through the list of Objects in the left of the window.

Object details



For each selected Object, the text in the base language will be shown in the top section of the window and the entries requiring translation are shown in the bottom section.

If the translated description is longer than the base language description the translation will be highlighted in red.

Save ■ your translations periodically as a safeguard.

4. Save your translations

If you attempt to Close the file without Saving first your translations will be lost. You will be warned if there are any unsaved changes and given the option to save them.

- 5. Zip the *Optionally zip* the tl*.txt translation files. translations
- 6. Return translation files

The generated zip file tlfiles.zip or unzipped tl*.txt translation files should be returned for incorporation into the development environment.

Integration of the files back into the development environment is described in

14.3 Load Translations for Appropriate Language.

14.3 Load Translations for Appropriate Language

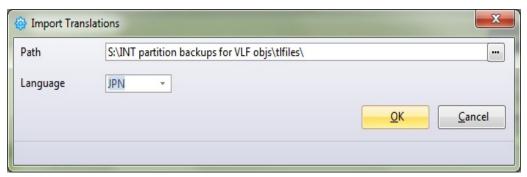
Step How to do it

- 1. Receive The translator will return a tlfile.zip file or a group of tl*.txt files translations for processing and integration into the repository.

 back from the is received, unzip to extract the associated tl*.txt files. Ensure the file TLSRCCE.txt is included with returned files (or use translator the original from the sent Version).
- 2. Load the Select the *Import Translations* from the Tools ribbon to import the translations translations returned after translation work has been completed.



Enter the location of the tl*.txt files and the language of the translations to be loaded back into the LANSA internal database.



3. Verify The translations will be loaded back into the active partition. translations Check the object definitions to verify that translations for the loaded selected language have been correctly loaded into the active partition.

15. Review your Deployment Package

After you have built and distributed a Package you may need to review the contents and definition of the Package. There are several ways to do this:

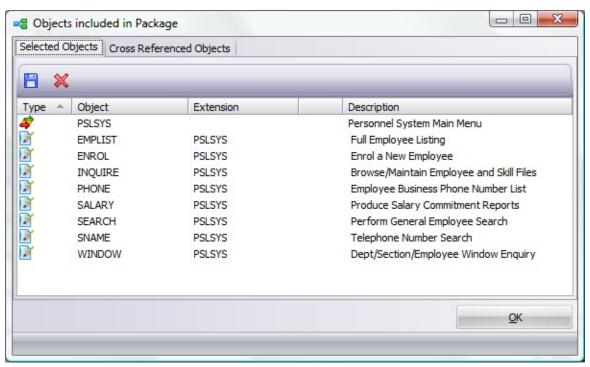
- If you used Cross-referencing in conjunction with your Object Selection you can review the full list of objects included in the Package using the 15.1 Objects included in Package.
- To search one or all Applications to check if an object was included in the Package use the 15.2 Search Application feature from the Options Menu.
- If you want to review the options you will need to create a new version or patch.

15.1 Objects included in Package

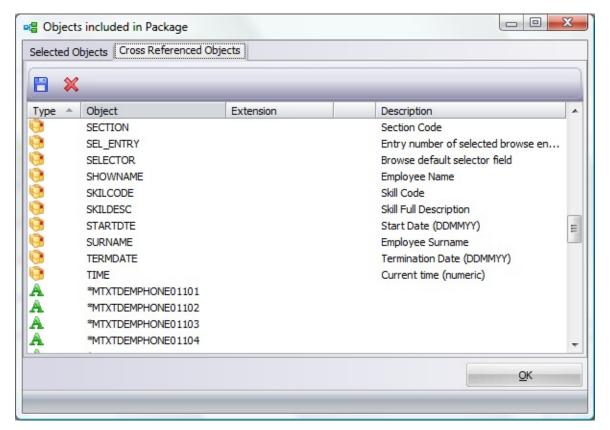
After a Package has been Built select the Package in the Main Window, then use the Included Objects tool on the toolbar or the Objects option from the Package menu.

During the Build step, the repository objects included in the Package definition are used in combination with the cross-reference settings to determine the complete set of objects to be included in the Package.

The first tab in the Object included in Package dialog, Selected Objects, details all the objects specifically selected for inclusion in the Package.



The second tab, Cross Referenced Objects, details all objects included in the Package, that is objects specifically selected for inclusion in the Package and any cross referenced objects included in the Package when it is built.



Use the Save to File option on the Selected Objects tab or Cross Referenced Objects tab to create a text file of the objects.

The Delete option \bowtie can be used remove objects from the Package. If any objects are removed from the Package, the Package will need to be re-built.

When a Package is modified and saved, any cross-referenced objects previously included in the Package will be removed. The cross-referenced objects will be regenerated when the Package is re-built.

If you re-build a Package (without re-saving the definition) you have the option to:

re-build the list of cross-referenced objects

or

• use the cross-reference object list generated during the last build.

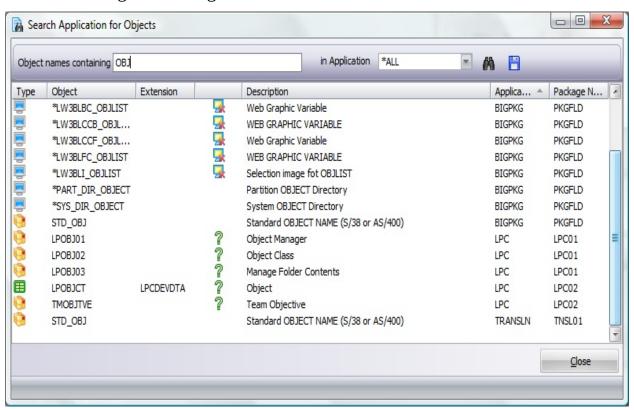
If any of the objects associated with the Package have changed it is recommended that you rebuild the cross-referenced object list.



15.2 Search Application

In the main window, select Search Application from the Options Menu.

Use Search Application to check if an object was previously deployed, and if so in what Package or Packages.

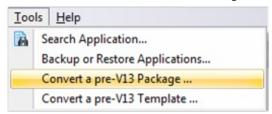


To commence a search, enter any string you want to search for in the object name then nominate whether to search all applications or a specific application. Press the search button to start the search. Result displayed in the list can be saved to a text file by using the Save button.

Initiate a new search by changing the search criteria and pressing the search button again.

15.3 Convert pre-V13 Package

If you have a complex package defined in a pre-V13 LANSA environment, this package can be converted and used as the basis for a new application in Version 13. In the main window, select Convert a pre-V13 Package from the Tools menu to locate and convert a package defined in an earlier version of LANSA.

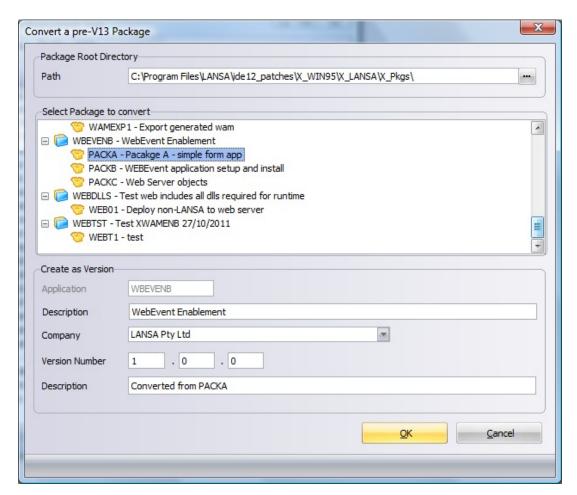


If the previous application consisted of multiple packages, whether designed to be installed to separate locations (e.g. separate client and server packages) or consecutively (e.g. LANSA execution environment in one package and Application objects in a second package), select the package which includes the most complexity for conversion. Any additional information can be added to the new Version after the package conversion is complete.

Important: Unless your pre-V13 Package includes complex commands, a large selection of objects or other complexity which would be time consuming to reenter it may be easier to bypass the conversion and build a new version from scratch using an appropriate template. This will make it easier to maintain as the template will restrict the options available in the user interface.

The conversion interface is a four step process:

- 1. Enter the path where the package to be converted exists.
- 2. Select a package to convert.
- 3. Confirm details and enter an appropriate Company.
- 4. When details are complete select OK to convert the selected package.



Path Start by selecting the location of the package to be converted.

This should be in the format ...

 $X_WIN95X_LANSAX_PKGS$

Select Package to Convert

If the Package Root Directory entered contains any Package definitions a tree of the Application and Package will be displayed.

Select the Package to be converted. Details from the selected Package will be used to pre-fill the Version details.

Application Application is pre-filled from the copied Application and cannot be modified.

Description Enter an Application Description. This Application description will be used when installing and uninstalling the Application.

Refer to Create an Application for more information.

Company Required.

Select a Company to be associated with this Application. To define a Company use the ellipse button to access the Company Maintenance. The Company and associated details are used during the build processing as well as during installation and uninstallation of the Application.

Refer to Set up Company Details for more information.

Version Number

Enter a Version number. This will be the first Version of this Application built using the new Deployment Model. Refer to The

Deployment Model for more information.

Description Enter a Package description or use the pre-filled description.

16. Test your Deployment Package

- 16.1 Allocate a PC for building your Deployment Packages
- 16.2 Allocate the necessary PCs for testing your Deployment Package
- 16.3 Troubleshoot the Deployed Application

16.1 Allocate a PC for building your Deployment Packages

Note:

- Do not deploy your application objects directly from a developer's PC.
- Do not use network clients for building and deploying packages.
- It is strongly recommended that you do not install the application on the machine that constructs it.

Dedicate a machine to assembling and building packages. This machine must not be used for developing or executing the application being deployed. A dedicated machine allows you to test and rebuild Versions and Patches without inadvertently including any application modifications.

On the dedicated machine, install a full version of LANSA. LANSA refers to this as the Deployment System.

When development has finished, ensure that each finished application is moved to your Deployment System. You can then construct, or reconstruct the package from a controlled environment without the risk of including unexpected changes.

Also see

16.2 Allocate the necessary PCs for testing your Deployment Package

16.2 Allocate the necessary PCs for testing your Deployment Package

Always test your Package before distributing it to customers. This can be a time consuming task but is essential to ensure a problem free installation.

Test your deployment Package by installing it onto any PC configuration you intend to support.

Consider:

- testing on different operating systems
- including a PC with your minimum specification
- installating in different languages (if required)
- trying out variations on the SETUP values which are available to be modified during installation
- testing variations of the X_START values available when the application is launched
- testing with different DBMS types Sybase ASA and MS SQL Server, as applicable
- checking any user authorization that may be required (both for the deployed objects and the deploying environment)
- testing installation of your Package from the medium or location it will be distributed to customers from (e.g. from a central server location or on CD-ROM).
- installing on a network client.

The more flexibility you include in your Package by prompting for X_START and installation values, the more testing you will be required to perform to ensure the Package is robust and complete.

16.3 Troubleshoot the Deployed Application

After the deployed application has been successfully installed or updated, if you encounter issues when attempting to run the application, refer to the x_err.log as you would with any other Visual LANSA application. Specifically check for warning or fatal errors.

The x_err.log file is automatically generated and updated when an error occurs, both in the development environment or when running a LANSA application.

Some useful information about the x_err.log

- All Visual LANSA applications will produce an x_err.log when an error occurs.
- The x_err.log generally exists in the user's temp directory. However, depending on a user's PC configuration, it may reside elsewhere. The best method of finding the x_err.log is to search your hard drive.
- New entries get appended to the end of the x_err.log file. Over a long period, this can cause the x_err.log file to become quite extensive. It is recommended to periodically delete the x_err.log file to remove old entries.
- When using the Deployment Tool, the x_err.log file will only be found if a LANSA application error occurs during Package creation. Messages pertaining to the Package creation are logged in a separate file lpcreate.log. Refer to Package Build Log for more information.
- In a Client/Server environment, if the application is being run on the client when an error occurs, any associated information will appear in a Fatal Error dialog box. If however, the X_RUN is running on a server (via a CONNECT_SERVER), the server cannot initiate a dialog on the client. In this case the details will be logged into the x_err.log file.

Refer to Troubleshooting in the *Visual LANSA Administrator's Guide* for more information.

17. Install an Application

After defining and building a Version or Patch, it is ready to be distributed for installation.

There are three mechanisms used to install Versions or Patches with the most used options being:

Windows Installer File Install

Just in Time Install

If you encounter issues during the installation refer to 17.4 Troubleshoot the Installation.

LANSA Import can also be used to install definitions into a LANSA development environment.

Note: A Windows Installer file will not be generated if you have nominated to *Deploy to the Development Environment*. Refer to LANSA Import for instructions on how to install definitions into a Development Environment.

17.1 Windows Installer File Install

When you successfully build a Version an MSI file will be produced. This will include all the objects required to install your application whether it be a server installation, a client installation, a network client or other variation.

Similarly if you build a Patch, an MSP file will be produced. The MSP must be installed on top of an Application which has previously been installed using an MSI file. In the case of a Patch install, only new or modified objects will be installed. Any unmodified objects included in the Patch will not be reinstalled. Whether an object has been modified is determined based on the DLL File Version Number of the shipped DLL. Refer to Set DLL Version Information for more detail.

The products of the Deployment Tool build process, the MSI or MSP file, are automatically associated with the Microsoft Windows Installer process (msiexec.exe). To start the installation, you can:

- 17.1.1 Double-click the MSI or MSP file in explorer or
- use the 17.1.2 msiexec.exe Command-Line Options.

All Deployment Tool installations automatically log their installation behavior to an MSI*.log file in the end user's defined %TEMP% folder, for example: MSI2ce4c.log.

As well as installing the Application, a standard Windows program menu folder will be created for each Application using the Company and Application descriptions as folder names. This means that every Application built with the same Company details and installed on the same PC will be grouped together.

17.1.1 Double-click the MSI or MSP file in explorer

Step	How to do it
1. Deploy the Package	Copy the generated MSI or MSP file to the deployment media or appropriate location.
	By default the Application is built at <sysdir>\X_APPS</sysdir>
	The Version or Patch installer file can be found below this:
	<sysdir>\X_PKGS\<application>\</application></sysdir>
2. Install the Application	Open Windows Explorer and locate the Windows Installer File for the appropriate language.
	Double click .msi or .msp file to execute it.
	The application will now install. Follow the prompts and provide any installation details required. When the install is complete the Application can be automatically started or it can be launched from the desktop.

17.1.2 msiexec.exe Command-Line Options

When installing from a command-line, supply the MSI file name as a parameter to the msiexec.exe command at a command prompt. The general format of the command is:

msiexec /Option <Required Parameter> [Optional Parameter] [Visual LANSA Public Properties]

The Visual LANSA Public Properties are known to msiexec as Public Properties. The names of these properties are displayed to the right of the input boxes in the installation dialogs. For example **(DBII)** to the right of Data Source Name in *Setup the Local Database* dialog.

These Public Properties have the same name as x_run parameters so they can be easily matched up. All values are specified the same as for x_run except for boolean values. For x_run a Boolean is specified as Y for true and N for false. For an msiexec public property, 1 is true and 0 is false. Internally the public property value is mapped to what x_run expects. In fact when the MSI is built in the Deployment Tool mapping is the other way – from Y/N to 1/0. That's what an MSI requires. For example, x_run uses SUDB=Y and msiexec needs SUDB=1.

Also note that the parameters specified in the JIT setup.txt customisation file are Public Properties. So you need to specify SUDB=1 in this setup.txt file.

msiexec.exe has additional parameters and switches to alter its behavior. These include allowing unattended, quiet and verbose modes of installation. The logging level can also be modified at parameter level.

One of the optional parameters is to specify public property values. The LANSA-specific public properties which may be set on the command line are displayed to the right of entry fields in the installation dialogs in brackets, for example, **(DBII)**. These codes may be specified on the command line to set the Deployment Tool application field values such as DBII='My App DSN', DBSV=SRV1\SQLEXPRESS and DBAS=ACMEDB.

There is one LANSA public property which is not displayed on a dialog as it's a standard Windows Installer dialog. That is the installation Destination Folder. The name of this public property is **APPA**.

Note that when an MSI or MSP file needs to be specified, sometimes the full path is required. So it is recommended that you always specify the full path.

```
Install Options
      </package | /i> <Product.msi>
      Installs or configures a product
   /a <Product.msi>
      Administrative install - Installs a product on the network
   /j<u|m> <Product.msi> [/t <Transform List>] [/g <Language ID>]
      Advertises a product - m to all users, u to current user
   </uninstall | /x> < Product.msi | ProductCode>
      Uninstalls the product
Display Options
   /quiet
      Quiet mode, no user interaction
   /passive
      Unattended mode - progress bar only
   /q[n|b|r|f]
      Sets user interface level
      n - No UI
      b - Basic UI
      r - Reduced UI
      f - Full UI (default)
   /help
      Help information
Restart Options
   /norestart
      Do not restart after the installation is complete
   /promptrestart
      Prompts the user for restart if necessary
   /forcerestart
      Always restart the computer after installation
Logging Options
   /I[i|w|e|a|r|u|c|m|o|p|v|x|+|!|*] < LogFile >
      i - Status messages
      w - Nonfatal warnings
      e - All error messages
      a - Start up of actions
      r - Action-specific records
      u - User requests
      c - Initial UI parameters
      m - Out-of-memory or fatal exit information
```

```
p - Terminal properties
      v - Verbose output
      x - Extra debugging information
      + - Append to existing log file
      ! - Flush each line to the log
      * - Log all information, except for v and x options
   /log <LogFile>
      Equivalent of /l* <LogFile>
Update Options
   /update < Update1.msp>[;Update2.msp]
      Applies update(s)
   /uninstall <PatchCodeGuid>[;Update2.msp] /package <Product.msi |
ProductCode>
      Remove update(s) for a product
Repair Options
   /f[p|e|c|m|s|o|d|a|u|v] <Product.msi | ProductCode>
      Repairs a product
      p - only if file is missing
      o - if file is missing or an older version is installed (default)
      e - if file is missing or an equal or older version is installed
      d - if file is missing or a different version is installed
      c - if file is missing or checksum does not match the calculated value
      a - forces all files to be reinstalled
      u - all required user-specific registry entries (default)
      m - all required computer-specific registry entries (default)
      s - all existing shortcuts (default)
      v - runs from source and recaches local package
Setting Public Properties
   [PROPERTY=PropertyValue]
```

o - Out-of-disk-space messages

A help dialog is displayed by *msiexec.exe* if no options are specified on the command-line or if an error is encountered when processing the command-line. Sample installation commands:

```
Install a Version msiexec.exe /i product>.msi
Administrative install of msiexec.exe /a product>.msi
a version
```

Install a Patch msiexec.exe /p product>.msp

Install a Patch with msiexec.exe /p c:\product>.msp SUDB=1

database updates

Install to a specific msiexec.exe /I product.msi>

directory silently APPA=c:\MyInstallLocation /passive

LANSA Public Properties not documented on the installation dialogs

EXITDIALOGOPTIONALCHECKBOX Disable the Exit Dialog start check

box so that the application is not started. If it is set to 0 the check box is not shown. Also note that the /passive command line option and other options that cause the Exit Dialog not to be shown, will not

start the application either.

For additional documentation about the command line syntax, refer to the Windows Installer online resources.

17.1.3 Administrative Install

If running the installation with User Account Control (UAC) enabled, tasks which require administrator privileges include:

- Executing an *All Users* installation
- Updating a local database

To perform an elevated installation, that is to install applications or patches using Windows Installer on behalf of a user who is not a member of the Administrators group, the installation must be started from a privileged command prompt.

If privileges are required to install an application, subsequent Windows Installer reinstallations, repairs or uninstalls of the application must also use elevated privileges.

For more information on UAC refer to relevant Microsoft documentation.

To access a command prompt with administrator rights, select *Run as Administrator* option from the *Command Prompt* menu.

From the *Command Prompt* window execute the msiexec.exe command-line with the required parameters. For install, re-install and repair use the /**I** option. For uninstall use the /**x** option. If Programs & Features is used to uninstall then some directories and files will not be removed which may be removed manually (this will require elevation). All other changes to the system, for example to the registry, will still be backed out.

If the Windows Installer is executed under UAC but is not indicated as an Administrative install, the install will run as a *Limited* installation preventing an *All Users* install.

17.1.4 Installation from a Web Browser

As the build process generates a single MSI file it is simple to launch the Windows Installer file from a URL.

Click the link to download the MSI file and install Application.

If the MSI file has been signed by a validated digital certificate the following warning will be bypassed.

17.1.5 SCCM Integration

As the Visual LANSA Deployment Tool produces standard Windows Installer MSI and MSP files, these files may be integrated into Microsoft Systems Centre Configuration Management or Systems Management Server tools. Both of these products provide a sophisticated facility to distribute application software within a corporate environment and provide an alternative facility to LANSA's Just In Time feature.

17.1.6 Version Installation Dialogs

The dialogs displayed when installing a Version (MSI file) will depend on the Options and Settings selected when creating the package and the install msiexec.exe parameters used when running the Windows Installer.

A Patch (MSP File) only displays one configuration dialog which allows install behavior to be modified. That is the Database Setup dialog and the only option that may be enabled is the Setup Database checkbox. For all but one user of the database they must not check this box. Only one user must setup the database. This is extremely important.

It is possible to run the Windows Installer without displaying the following dialogs if the install is executed in quiet or passive mode:

/quiet Quiet display option.

The Windows Installer will run the installation without displaying a user interface. No prompts, messages or dialog boxes will be displayed to the user. In this mode the user cannot cancel the installation.

/passive Passive display option.

The Windows Installer will display a progress bar to indicate that an installation is in progress however there will be no prompts or error messages are displayed to the user. In this mode the user cannot cancel the installation.

Refer to msiexec.exe resources for more information.

Welcome Dialog

This is a standard Windows Installer Welcome Dialog using the Application description to identify the Application. The Application Name, for example D20A, is important to some aspects of the application but the end user will usually identify the Application by its description.

Press *Next* to proceed through the install wizard.

End-User License Agreement Dialog

A default *End User License Agreement* is supplied in the directory ... $X_AppsX_WixContent\license.rtf$. This is provided as an example only and should be modified to reflect your own licensing agreement before the Version or Patch is built.

Alternatively an Application and/or Language specific version of the License Agreement can be supplied in the *Files with Special Processing*. Refer to Files with Special Processing for more information.

Accept the Licensing Agreement terms and press *Next* to proceed through the install wizard.

Installation Scope Dialog

The Installation Scope dialog allows the user to choose to install the product for *All Users* or for the *Current User*.

By default, an end user is not afforded *privilege* status when operating under UAC conditions and therefore will only be able to install for the *current user*. This will also restrict their ability to install or modify a database during the installation process.

To perform an *All Users* installation when operating under UAC conditions the Application must be installed from an administrative command prompt. Refer to Administrative Install for more information.

Select the appropriate option and press *Next* to proceed through the install wizard.

Destination Folder Dialog

The Destination Folder dialog uses the Application description and Company description to construct a default location where the application will be installed.

- For an *all user* installation, Windows Installer will automatically pre-pend the defined *Program Folder*.
- For a *current user* installation, Windows Installer will automatically prepend the current user's virtual store.

The derived Destination Folder can be modified by entering the preferred location or browsing for it. When the Application is installed:

- **Special Files** will be installed in the **User** sub-folder and developer-defined shortcuts will use this location as their target.
- **X_Start** files will be created in the **Execute** sub-folder using the Application code as an extension. These files will be associated with the Application and double clicking them in Explorer will start the application.
- Any Network Client MSI file will be installed in the **Execute** sub-folder.

Enter the appropriate destination folder and press *Next* to proceed through the install wizard.

Setup Application Dialog

This dialog will only be displayed if the option *Display Application Dialog* is selected in the Deployment Tool options.

The values provided on this dialog, and whether they can be modified is determined by the options selected in the Deployment Tool interface when defining the Application.

Any value entered in the *Startup Form*, *Startup Process* or *Startup Function* indicates the DLL to be used to launch the application on completion of the wizard. Only one of these 3 options should be entered.

Refer to Required Execution Parameters for more information.

Confirm or modify any values as required then press *Next* to proceed through the install wizard.

Setup Local Database Dialog

This dialog will only be displayed if the option *Display DBMS Dialog* is selected in the Deployment Tool options. This dialog is only relevant when installing Applications with a windows database.

By default the Data Source Name will be set to the Application description in the appropriate language.

If the option *Setup Database* is not checked database additions and modifications will be bypassed by the install processing. Most installations which use a database directly would check this option. Exceptions are instances where multiple installs access the same shared database. In such a case it is up to the installer to check this option in the install dialog for the server and leave it unchecked for the clients.

The *Setup Database* and *Upgrade V12 Database* options will only be enabled for privileged installation when operating under UAC. This will require an Administrator to install the first instance of the Application to set up the database. Non-privileged users can then install the non-database components of the Application locally or on their own PC accessing this database.

Also note the field start file codes to the right in brackets for example, (DBII). As well as standard msiexec.exe parameters and switches such as '/i' and '/q', these code may be specified on the command-line to set the Deployment Tool application field values such as DBII='My App DSN',

DBSV=SRV1\SQLEXPRESS and DBAS=ACMEDB.

The values to the right of the input boxes are only displayed when SHOWCODES=1. They indicate the X_RUN or special values which can be used to set the corresponding parameter when executing the installation from a command line using msiexec.exe. For example, msiexec.exe /p c:\product>.msp DBAS=LANSADB
SUDB=1

Confirm or modify any values as required then press *Next* to proceed through the install wizard.

User ID Dialog

This dialog is displayed when one of these LANSA Features is installed: IIS Plugin, Web Support, or Java Service Manager.

The Public Properties on this dialog have the following names and properties: CREATENEWUSERFORSERVICE - New Installs default to Use Existing User. Upgrades default to Do Not Configure.

USERIDFORSERVICE

PASSWORDFORSERVICE

Note that none of these properties are remembered between Install and Upgrade. For Upgrade it is presumed that the user has already been created and therefore does not need to be done again.

Web Sites for IIS Plugin Dialog

This dialog is displayed when the LANSA Features IIS Plugin is installed.

The Public Properties on this dialog have the following names and properties:

WEBSITEFORIISPLUGIN - formatted as <length><website><length><website> For example, 016Default Web Site010Web Site 2

This value sets the right hand list box - WebSiteSelected. Only valid web sites will be listed.

WEBSITEFORIISPLUGIN is derived from WebSiteSelected when the Next button is clicked.

Web Site Virtual Folders Dialog

This dialog is displayed when one of these LANSA Features is installed: Web Images or Java Service Manager.

The Public Properties on this dialog have the following names and properties:

VIRTUALWEBSITE

VIRTUALDIRECTORYALIAS

Communication Ports Dialog

This dialog is displayed when one of these LANSA Features is installed: Listener, IIS Plugin, or Java Service Manager.

The Public Properties on this dialog have the following names and properties:

HOSTROUTEPORTNUMBER

JSMPORTNUMBER

JSMADMINPORTNUMBER

Other New Public Properties which have no User Interface and therefore default to these values unless altered on the command line:

HTTPPORTNUMBER - 80

AUTOSTARTJSMADMINISTRATORSERVICE - True

Setup Server Dialog

There are two dialogs relating to the server setup. These dialogs will only be displayed if the option *Display SuperServer Option 1* and *Display SuperServer Option 2* respectively are selected in the Deployment Tool options.

The values to the right of the input boxes are only displayed when SHOWCODES=1. They indicate the X_RUN or special values which can be used to set the corresponding parameter when executing the installation from a command line using msiexec.exe. For example, msiexec.exe /p c:\product>.msp DBID=*AS400
PSLU=LANSA99

Confirm or modify any values as required then press *Next* to proceed through the install wizard.

This dialog will only be displayed if the option *Display SuperServer Option 1* is selected in the Deployment Tool options. This dialog is only relevant when installing Applications which require X_RUN to create a SuperServer connection to the database.

This dialog will only be displayed if the option *Display SuperServer Option 1* is selected in the Deployment Tool options. This dialog is only relevant when installing Applications which require X_RUN to create a SuperServer connection to the database.

Setup Type Dialog

Currently there is no difference between a *Typical*, *Custom* or *Complete* installation, however this feature may be utilized in the future releases of LANSA.

Press *Next* to proceed through the install wizard.

Verification Dialog

After entering or confirming the values on the previous dialogs a final dialog is displayed to highlight any obvious issues and allow a final confirmation of details before the install commences. Several checks are performed, based on values entered, including the existence of the installation location and DSN and a check to ensure the database server can be accessed.

If any validation fails the *Install* button will be disabled preventing the installation from continuing. Review the text information and use the *Back* button to return through the dialogs to correct any values, or modify the external environment to rectify any issues.

Press *Install* to start the installation process.

Progress Dialog

This is a standard Windows Installer MSI dialog which displays the progress of the installation.

Press *Cancel* to interrupt the installation process.

Completion Dialog

After the installation has completed successfully, the following dialog is displayed.

At the base of the dialog is an option to automatically start the application. This will launch the form, function or process nominated in and described in Setup Application Dialog. An application shortcut will be created on the desktop.

Press *Finish* to finalize the installation process and optionally launch the Application.

17.1.7 Patch Installation Dialogs

After an Application has been installed using an MSI file, it can be patched by a compatible MSP file.

As a Patch install does not support the full functionality of a Version install, the number of dialog is considerable reduced and do not allow any input related to configuration.

Note: Versions consist of a three part number, for example, 1.22.333 and Patches use a four part number, for example, 1.22.333.4444, which relates to the Version the Patch is based on. It is also important NOT to rebuild a Version after it has been released and installed as any subsequent Patches will be incompatible unless they are also rebuilt and re-released using the set of GUIDs generated for the Version. Microsoft Windows Installer technology will enforce this relationship.

Welcome Dialog

This is a standard Windows Installer Welcome Dialog using the Application description to identify the Application.

Press *Next* to proceed through the install wizard.

Confirmation Dialog

Installing a patch is a simple matter as it only requires the end user to answer a question if the application directly uses database tables. That question is whether or not to setup the database. For all but one of the users of the database the checkbox should be left unchecked - the default.

The patch's relationship to the primary application version and the primary's current configuration is all that is necessary. Once a patch has been started, following the same steps as described for an application installation, the end user simply has to confirm that they wish to proceed.

Patches may deliver a wide variety of changes to an Application, generally this will involve updating '.dll' and '.exe' files and modifying shortcuts and so on.

A patch is capable of including changes to a database, for example, table definition modifications. Uninstalling database changes require careful consideration as the default behavior is to restore the database table back to its previous state but keep all the data in its current state, apart from the dropping of new columns that were added by the patch. Some complex database changes may allow, say, larger numbers to be in a column, so when you restore the previous column with the smaller number limits with the larger number the restore will fail. Needless to say, before installing a Patch that contains database changes a backup should be made and also if the Patch needs to be uninstalled a backup should be made before doing so.

Press *Next* to proceed through the install wizard.

Progress Dialog

This is a standard Windows Installer MSI dialog which displays the progress of the installation.

Press *Cancel* to interrupt the installation process.

Completion Dialog

Confirmation that the Patch has been successfully applied is conveyed by the completion dialog.

Press *Finish* to finalize the close the wizard.

17.1.8 Network Client Installation

The main Application installation is generally referred to as the *primary installation* to distinguish it from the *network client* installation which is a small MSI file automatically created for most *primary installations* and delivered by the *primary installation*. A *network client* installation will only install shortcuts to the *primary installation* on a file server from where it must be executed.

To install a *client application* in a corporate environment, you must first install the *primary application* on a file server. Then provide a shortcut to the *network client* installation MSI file to users with access to that file server. For example: \\SRV1\Programs\ Lansa Pty Ltd\Personnel

System\X_Win95\X_Lansa\Execute\D20A_Client_En-us.msi

This introductory dialog, which also displays progress information, is the only dialog presented for a *network client* install. No configuration options are required as the client uses the *primary installation* as the target for any shortcuts created. These shortcuts must be defined by the *primary installations* definition.

17.1.9 Change, Repair or Remove Application

An Application can be changed, repaired or removed by

- re-executing the MSI file used to install the Application
- using the standard Windows Program and Features dialog

Note: Currently the **Change** function will execute but there are no options available to change. In future releases optional features, like installing documentation, may be implemented.

Re-install MSI File

Once an application has been installed, re-executing the MSI file will present the end user with a maintenance dialog. This allows the Application install options to be changed or the Application can be repaired or uninstalled. If the MSI file has been rebuilt since the initial installation of the Application, the Application must be uninstalled using the Windows Program and Features interface then the new Version can be installed with any modifications required.

Windows Programs and Features

Alternately use the Windows Programs and Features dialog to:

- Double click the Application to remove the Application
- Use the right click menu to select Change, Repair or Remove the Application

The Application description, Company description and Version number are used to identify the Application in the list of programs. If the Company information in the Deployment Tool included URL details, these URLs are available when the application is selected. This can be a useful mechanism to link to a website for access to upgrades.

Simply double click the Application entry to commence the uninstall.

17.2 Just in Time Install

Just in Time (JIT) deployment is a centralized, automated update facility for deploying Packages set up to use Just in Time processing.

All the Packages associated with an Application are installed on a centralized Application Server, that is, the initial Version and any subsequent upgrade Versions or Patches.

Just in Time installation needs to be set up when the initial Application Version is installed. This initial Version needs to be applied to the target PCs using the MSI.

After the initial installation, each time a user launches the Application from a Target PC, the Target PC connects to the Application Server to see if new upgrades are available to install. Any new upgrades will be automatically installed before the Application is launched.

This option can greatly reduce the administration costs associated with deploying application upgrades.

Refer to Just In Time Upgrade Guidelines for more information.

17.3 LANSA Import

Deployment Packages created with the Package setting *Deploy to Development* System can be imported directly into another LANSA development environment using the standard import option.

This is a mechanism for moving LANSA internal definitions between LANSA systems.

It does NOT:

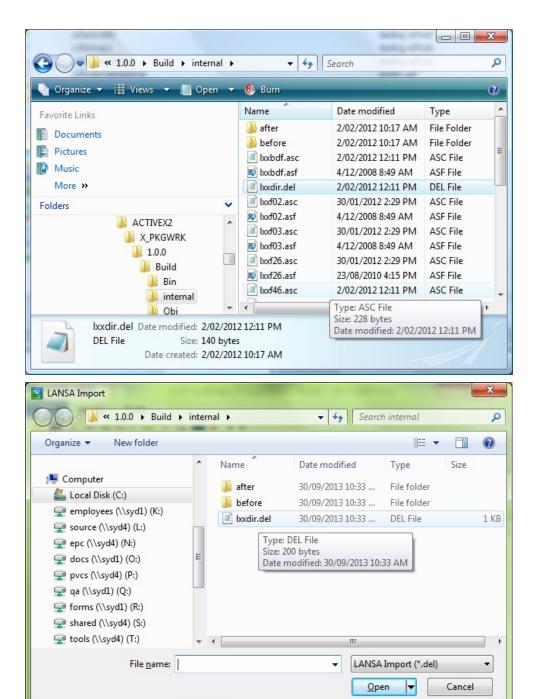
- Execute the Application installation
- Deploy DLLs
- Include runtime environment
- Process any installation options or settings

do it
0

- 1. Log on to the LANSA Development environment
- Log on to your LANSA Development environment taking care to select the partition where you intend to import the Package.
 - Also select an appropriate Task to be used for the import.
- import

2. Locate and From the Tools ribbon select the Import option. This allows you to execute an import while within the LANSA development LXXDIR.DEL environment.

> Locate the Package folder then select the internal folder below the Package folder. This should contain an LXXDIR.DEL file created by the Deployment Tool. Select this as the file to import.



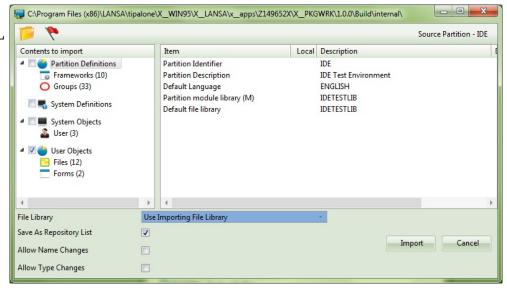
File Name

Specify the file name containing the import listing. By default, LANSA imports create an LXXDIR.DEL file that contains import listing.

Press the Open button to select the file to be imported.

3. Open and import LXXDIR.DEL

Review the import options and press Import to start the import.



When the import has completed review the import log for any errors.

17.4 Troubleshoot the Installation

MSI*.log

All Deployment Tool installations automatically log their installation behavior to an MSI*.log file in the end user's defined %TEMP% folder, for example: MSI2ce4c.log.

dpinstal.log

When a Package is installed using Windows Installer File Install or as a 17.2 Just in Time Install, the installation process creates a log file *dpinstal.log* under the directory ...\X_APPS\<application>\<Version or Patch>.

If there are any issues or errors encountered during the installation of your Application, check the dpinstal.log file. Specifically check for warning or fatal errors.

installtrace.txt

If interrogation of the dpinstal.log does not identify why your installation failed, an additional trace file *installtrace.txt* can be produced by setting the appropriate X_RUN values.

For an MSI install the X_RUN must include:

ITRO=Y

For host monitor package installs the X_RUN must include:

ITRC=ALL

The *installtrace.txt* will be generated into your temporary directory.

...%temp%.

This file logs the progress through the installation and may need to be forwarded to LANSA for interpretation.

18. Upgrade an Existing MSI Application

An end user cannot downgrade the current application, for example Version 1.22.444 cannot be upgraded to Version 1.22.333.

Follow the same procedure that is used to install a new version on top of an existing application. That is, simply double click the MSI file or run **msiexec.exe** from a command-line.

An upgrade defaults all settings to the same as the previous install. It will display the Local Database Dialog to allow the Setup Database check box to be selected as appropriate, if the dialog has been enabled. All other user data entry dialogs are skipped. So for an upgrade only the following dialogs may be displayed:

- Welcome Dialog
- License Agreement Dialog
- Local Database Dialog (optional)
- Verification Dialog
- Progress Dialog
- Completion Dialog

Refer to section 17.1.6 Version Installation Dialogs for more information about the above dialogs.

The upgrade process involves uninstalling the previous version of the application before installing the new version. The new version must contain the complete application including any changes to the database. The new installation will save a copy of the partition folder from the old version and restore it before running the upgrade process on the database, which for tables that change involves unloading the data, applying the changes and then reloading the data.

A new version of an application cannot be installed 'side-by-side' with a previous version, therefore installing a new version will always result in the previous version being removed first, leaving one entry for the application in the control panel.

Note 1: As an upgrade uses the values from the previous install and does not allow them to be changed (except for the database parameters). If an invalid value was used it will still be invalid. To override this behaviour, use a command line prompt with msiexec.exe to run the install and pass the correct parameter. For example, if the Form name in the MSI was originally

"MYFORM" but needs to be changed to "MYFORM2" then this is an example of the command line:

Msiexec.exe /i MYAPP_v1.0.2_en-us.msi FORM=MYFORM2

Note 2: If the error is discovered AFTER upgrading, you will need to uninstall and then re-install the application using the latest MSI file. Tables in the database will already have been updated so do not install the original MSI which may have different table definitions.

19. Upgrade an Existing V12 Application

By default, an MSI installs into \Program Files for an All User Install or \Documents and Settings for a Current User install. If an existing Version 12 LANSA application has been installed into an alternate location the MSI may not be able to install into the same location, for example, the default location c:\x_win95.

To upgrade a Version 12 LANSA application the LANSA Developer must identify that the MSI needs to offer the opportunity to upgrade. When the MSI is installed it will then prompt for the location of the existing install, offering the last installed application as the default. It will copy the whole partition across to the new location and then perform the install. In particular this will allow the database tables to be upgraded correctly.

To support this functionality there are two settings in an MSI:

Delete Previous Target Directory If this Application has already been installed on the target machine in a different directory, the install process can attempt to automatically remove the previous installation. The Application Code will be used to look up the previous install location in the registry and will attempt to delete the associated folder. This setting can be prompted during the installation.

Yes – Remove the directory where this Application was previously installed.

No – Do not remove the directory where this Application was previously installed.

Default: No

Previous If the previous installation is to be removed, indicate the name of Application the Previous Application Folder as it was installed. Any desktop icon or program folder matching this name will be deleted during

Name the installation process.

Directory Default: [Application Name]

Refer to Upgrade Options for more information.

20. Cloud Customizations

This section provides some ideas on how you may customize your Cloud environment to more easily administer your LANSA application.

- 20.1 Amazon Web Services
- 20.1.2 Procedure to follow when Upgrading Your Application
- 20.1.3 Using a Single Availability Zone (AZ)
- 20.1.4 Auto Scaling Considerations

20.1 Amazon Web Services

20.1.1 Add Alarms to Your CloudWatch Log Files Installation Error Alarm MSI installation error Other Alarms to Consider

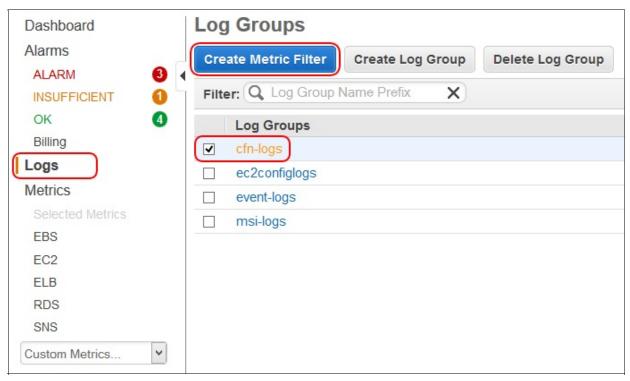
20.1.1 Add Alarms to Your CloudWatch Log Files

The standard LANSA CloudFormation template will create CloudWatch log files for you so you may query what is happening inside your EC2 instances without needing to RDP into the instance itself. You may also add alarms which are triggered on the contents of the log files. Two very useful log entries to look for is a general installation error and an MSI installation error.

In order to create an alarm you must first create a log group to apply it to. The simplest way to do this is to create a LANSA stack.

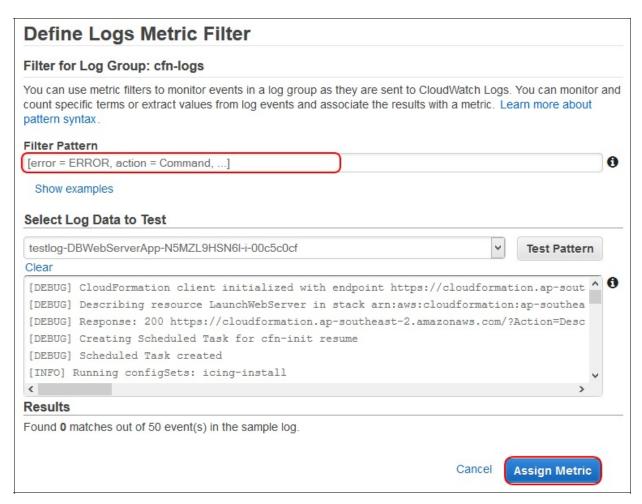
Installation Error Alarm

- 1. Open the *CloudWatch* dialog
- 2. Click on *Logs*, select the **cfn-logs** in the *Log Groups* and then click *Create Metric Filter*.

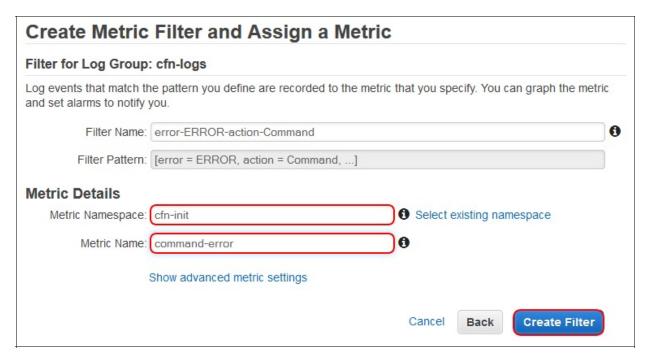


3. Use a space delimited *Filter Pattern* by surrounding the pattern in square brackets – [].

To detect errors when executing *CloudFormation* scripts you need to search for *ERROR* in the first field and *Command* in the second field, the remaining fields are not relevant so use ellipsis (...) to indicate that. If the sample log data contains your error log data you can click *Test Pattern* to test it. Otherwise just click *Assign Metric*.



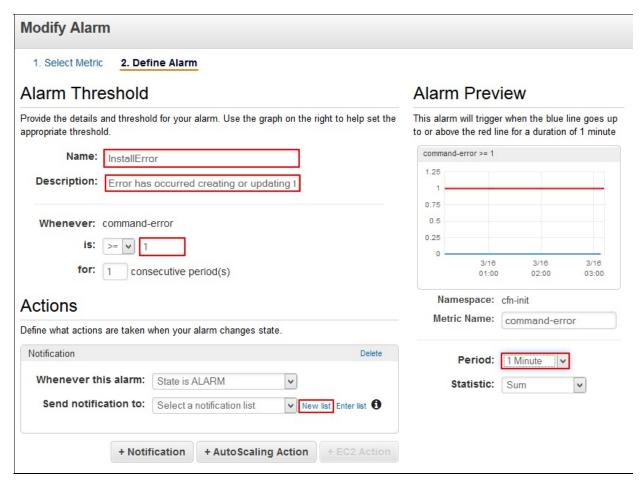
4. Change the default *Filter Name* if you wish and then enter the metric *Namespace* and *Metric Name* as indicated below...



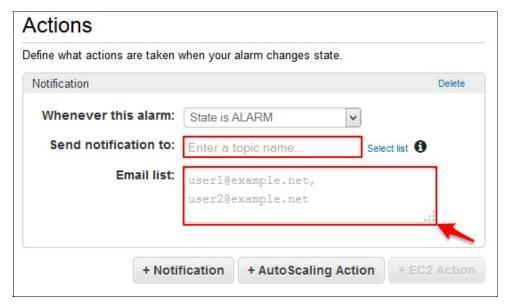
5. The following result will be displayed with the text that's highlighted in red. Click *Create Alarm*.



6. Enter a *Name* and *Description*. This description is "Error has occurred creating or updating the stack". The alarm needs to be displayed as soon as possible so it is set to a single error line in one period and the period is 1 minute. Then click on *New List* if you haven't already got a notification list



7. Enter the name of your topic like *LansaAdmin* and enter some email addresses separated by commas. Note that you may enlarge the email list by dragging the bottom right hand corner as indicated by the arrow. Then click *Create Alarm*.



- 8. This alarm is now ready to send an email to the list you created whenever there is an installation error in the LANSA Stack. There is one issue: the alarm will be listed under INSUFFICIENT. This is because the filter rarely matches any data at all. Only when there is an error. So another filter needs to be created that matches on all data but uses a Metric Value of zero so it does not affect the statistics.
- 9. So, create another filter on the cfn-logs Log Group and enter no pattern. Just click Assign Metric.
- 10. Enter the same Metric Namespace and Metric Name as for the previous filter, but set the Metric Value to 0.
- 11. The cfn logs update about once per minute and hence will always have data points.

MSI installation error

You may follow a similar procedure to <u>Installation Error Alarm</u> to add a filter to the event-logs Log Group. These Log Streams are an amalgamation of the three Windows event logs: System, Security and Application.

An MSI installation error is indicated by an *Error* being reported in the *Application Event Log* in the *MsiInstaller* system. The Filter Pattern to represent this is:

[EventType=Application, Status=Error, errorcode,system=MsiInstaller,computer,message]

Specify your metric details as you desire.

Create the alarm with the same settings as the Installation Error Alarm.

Also the Insufficient data alarm may be eliminated by creating a filter for all data the same as described in the Installation Error Alarm.

Other Alarms to Consider

Auto scaling is not a panacea for continually running your application in an optimum state. It is too slow to cope with massive spikes in load and adjust quickly enough and be efficient with resources. It is good at gradually scaling out as load gradually increases and the opposite, scaling in as load gradually decreases.

So you need to be warned when spikes in load are occurring. Some alarms to consider are

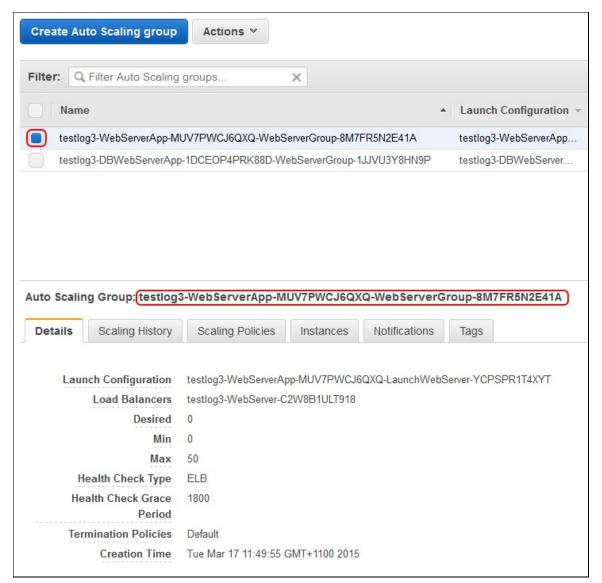
- EC2 CPU utilization sustained over 90% for maybe 5 mins and send an email
 - If expected to be sustained, manually add desired instances to Web Server ASG.
- RDS CPU utilization sustained over 90% for maybe 30 mins and send an email
 - If expected to be sustained, plan a time to scale up the RDS instance.
- Watch Load balancer error rate and web server error rates
 - Troubleshoot cause. Likely to be solved by more EC2 instances or scaling up the RDS.

20.1.2 Procedure to follow when Upgrading Your Application

In order to upgrade your application using a new MSI you will Update the LANSA Stack using the procedure described in the Cloud tutorial. If you use this method as described, your EC2 instances may be terminated by the Elastic Load Balancer because they fail to respond quickly enough to the health check. A new instance will be created in its place, but it lengthens the whole process.

The best way to ensure that no instances are terminated, nor new ones created, is to suspend all processing in both Auto Scaling Groups. This may only be achieved using the AWS SDK or command line. This example describes how to do it using the AWS command line. You will first need to install and configure it as described here: AWS Command Line Installation Guide.

Use the AWS console to obtain the names of your auto scaling groups. Navigate to the EC2 pages and then Auto Scaling Groups. Select one of your Auto Scaling Groups and copy the name from the *Details* tab sheet:



In a Windows command prompt type the following:

aws autoscaling suspend-processes --auto-scaling-group-name testlog3-WebServerApp-MUV7PWCJ6QXQ-WebServerGroup-8M7FR5N2E41A Do the same with the other Auto Scaling Group.

aws autoscaling suspend-processes --auto-scaling-group-name testlog3-DBWebServerApp-1DCEOP4PRK88D-WebServerGroup-1JJVU3Y8HN9P

Now you may upgrade the DB Web Server, followed by the rest of the Web Servers as described in the tutorial. The Auto Scaling Group will ignore all requests from the Elastic Load Balancer.

When all installs have fully completed (ensure that the CPU utilization has returned to nearly zero), you may resume processing in the Auto Scaling Groups:

aws autoscaling resume-processes --auto-scaling-group-name testlog3-WebServerApp-MUV7PWCJ6QXQ-WebServerGroup-8M7FR5N2E41A aws autoscaling resume-processes --auto-scaling-group-name testlog3-DBWebServerApp-1DCEOP4PRK88D-WebServerGroup-1JJVU3Y8HN9P

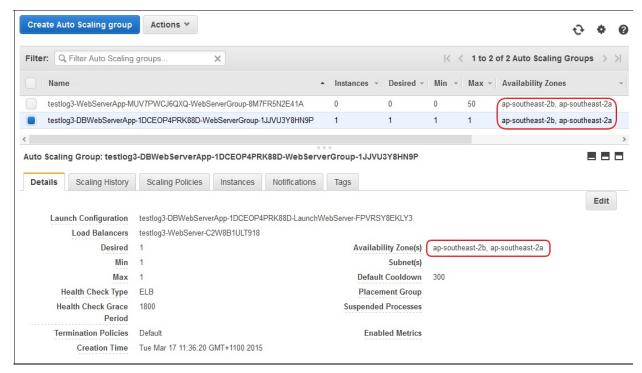
20.1.3 Using a Single Availability Zone (AZ)

The default stack created by the LANSA templates is to create EC2 instances in all the *Availability Zones* (**AZ**) in the current Region. Once the stack has been created you may easily alter it to use a Single AZ. Using multiple AZs provides better fault tolerance but this comes at the expense of slower access to the database server from the AZ which the database server is not in. When the Sydney Region is used, there are 2 AZs. Benchmark tests achieved a 20% speed increase. This is because the network speed between AZs is slower than the network speed within an AZ. So, you may choose to lower your costs by choosing to use a Single AZ instead. The costs are lower because you may require 20% less resource to be running to support a particular workload.

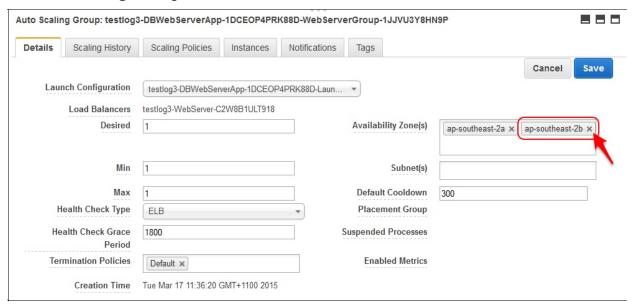
This issue is exacerbated in Regions which have more AZs. For example the Virginia Region has 4 AZs. In this case 3 out of 4 of the AZs are not the AZ of the database server. Hence 3 out of 4 (75%) transactions are slower, as compared to 1 out of 2 (50%) of the transactions in Sydney. Given that the redundancy provided by 4 AZs is probably unnecessary, AZs need to be removed. The same procedure below may be used to reduce the number of AZs from 4 to 2. When removing AZs from the Auto Scaling Group, leave 2 AZs rather than just the 1.

When you create the stack, do not specify extra web servers. Leave it set to the default of 0. Once the stack is fully operational you may make these changes. That is, use the application and test that it's working.

- 1. Navigate to the **RDS AWS** console page.
- 2. Locate the *DB Instance* and identify the *Zone* it is running in, say apsoutheast-2a. This is the **AZ** that all the EC2 instances will need to be created in.
- 3. Now navigate to the *Auto Scaling Groups* listed on the **EC2 AWS** console page. You'll notice that both *Auto Scaling Groups* are in multiple **AZ**s:



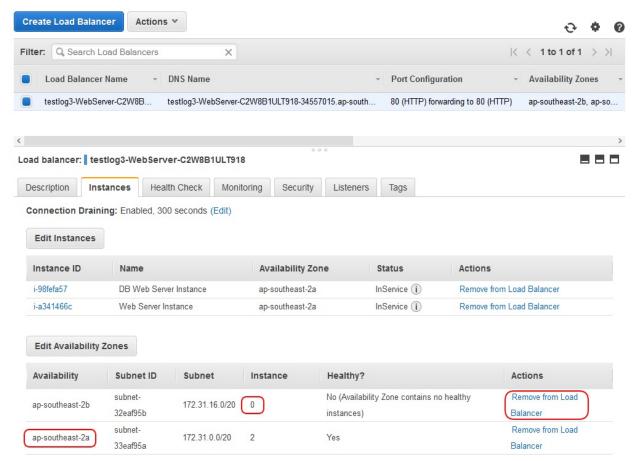
4. Choose the DB Web Server and edit the Auto Scaling Group details. Identify the AZ you need to remove (e.g. ap-southeast-2b) and click the 'x'. Save the Auto Scaling Group



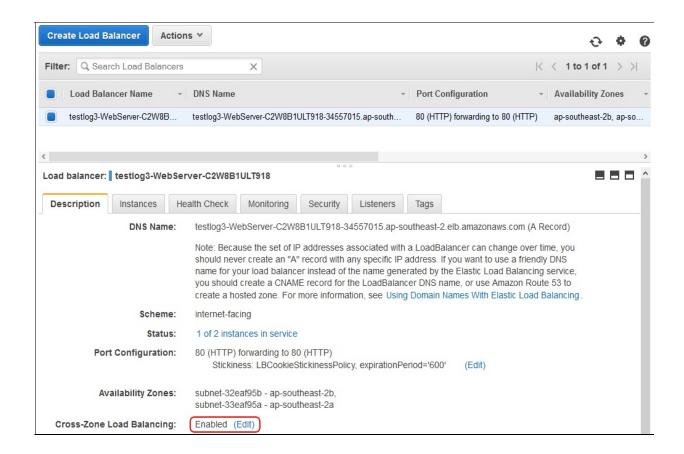
- 5. Check the instance tab. If the instance is in the correct AZ, for example **apsoutheast-2a**, then you may continue to the next step. Otherwise you will need to wait for a new instance to be created in **ap-southeast-2a** and for it to be fully operational before you may continue.
- 6. Do not continue if the DB Web Server instance is being re-created in the new

AZ.

- 7. Now modify the other *Auto Scaling Group* in the same manner. At the same time you may alter the **Desired**, **Min** and **Max** settings to create some web server instances in the single AZ.
- 8. Alter the *Elastic Load Balancer* by removing the unused AZ from the *Load Balancer*, in this case **ap-southeast-2b**. Click on the *Instance* tab. apsoutheast-2b will probably have 0 instances. If it doesn't, check that you have the correct AZ. It may be in the process of removing instances from it. Click the action *Remove from Load Balancer*:



6. Finally, alter the *Elastic Load Balancer* so that *Cross-Zone Load Balancing* is disabled:



20.1.4 Auto Scaling Considerations

These examples are mostly focused on triggering scaling events rather than working out real-world settings for when to scale up and when to scale down. This is because each application and its users create a unique scenario of variability of load and appropriate maximum response times. Some applications may be fine with maximum response times of 30 seconds or more provided the average stays below 2 seconds. Whereas others would want to ensure that never occurs and hence scale out much more quickly.

It would be very desirable to be able to demonstrate the auto scaling, but it takes 20 minutes to bring a web server alive and so a demonstration needs to take at least 60 minutes to show it scaling out and then scaling back. That would usually be impractical. The tutorials provide a manual demonstration of scaling that will need to suffice.

Firstly, use a large database instance that will cope with the maximum expected load as it cannot be scaled out without taking the whole system down. Thus the number of EC2 instances will control response time which the Auto Scaling Group (ASG) manages.

These tests were run in a Single Availability Zone (AZ) stack rather than the default Multi-AZ Stack. This should make no essential difference to the scaling process as the alarms are purely based on CPU utilization inside the EC2 instances which is not effected by accessing the RDS instance in another AZ. Yes, the average response time is slower but the amount of CPU used will be about the same, there will just be some sub-second delays in execution which will not change the per minute CPU utilization.

The scaling works on the average CPU utilization of the existing instances. Only the Web Server ASG is configured to scale out. If there are no existing instances in this ASG, there is no average CPU utilization and so scaling cannot occur. Thus there must be at least 1 instance in this ASG.

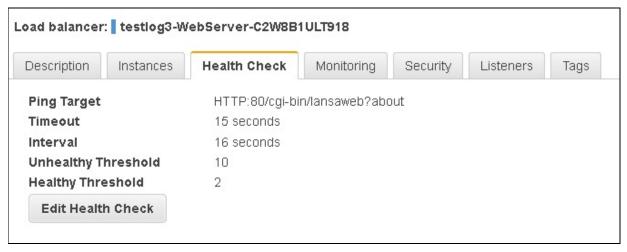
Why do we not just have one ASG? What is the purpose of having two? This is because updating the database must be managed by a single instance. There cannot be more than one instance responsible for changing tables otherwise the state of the database stored on the instance and the state of the tables in the RDS will not match. It's critical for table updates like adding a column that the web server matches the database server state. Hence there must only be 1 web server responsible for this. This is what the DBWebServer ASG exists for. To ensure that 1 and only 1 web server instance is running which is in control of changing

the database state.

In the CloudWatch AWS Console change the WebServerApp CPUAlarmHigh from >70 for 15 minutes to > 70 for 5 minutes. This will allow the scaling to occur more quickly. Also change the WebServerApp CPUAlarmLow from <30 for 15 minutes to <30 for 5 minutes.

Now change scaling policy for both alarms to wait 1200 seconds (20 minutes) before allowing another scaling event. This must be at least as long as it takes to get an instance up and running and also then to positively affect the CPU utilization.

Reduce the ELB health check so that an instance is recognized as Healthy as quickly as possible. The Timeout must be more than most response times. Preferably more than the maximum response time. In this case 15 seconds may keep the instance healthy whilst allowing scaling to occur ASAP. Make the Interval 1 second longer at 16 seconds, the Unhealthy Threshold to the maximum of 10 and the Healthy Threshold to the minimum of 2. This takes health out of the equation as much as possible to allow the ASG to scale as it becomes necessary without instances taking too long to become healthy nor be terminated too soon when under load.



A load test was then run on this LANSA Stack with 30 virtual users for 43 minutes. This scaled out from 1 to 3 instances and then once the load finished, gradually reduced back to 1 instance.

21. How To ...?

- 21.1 Create a Deployment Version
- 21.2 Connect to the Required Server
- 21.3 Search Packages including an object
- 21.4 Back up your Application Definitions
- 21.5 Deploy an external ActiveX control
- 21.6 Deploy LANSA Components published for ActiveX
- 21.7 Deploy User Defined RDML BIFs
- 21.8 Install Runtime License for Application
- 21.9 Set DLL Version Information
- 21.10 Specify Installation-Specific Database Connection Settings
- 21.11 Deploy a 64-bit Application
- 21.12 Install Application Files into a consistent schema name

21.1 Create a Deployment Version

This is the essence of what the deployment tool is designed to do. Simplistically, the process of creating the first Deployment Version in an Application from beginning to end is as follows:

The crux of successfully deploying an application is to Plan your Deployment, including the necessary objects and using the appropriate Options and Settings to define the Version.

21.2 Connect to the Required Server

How do you get the correct Server Name if your application has an Ancestor form that is inherited by all forms and does an automatic connection to the server using the CONNECT_SERVER Built-In-Function?

Note: The Routing Table lroute.dat is shipped with the Application.

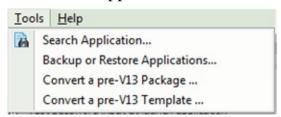
The options available to you are:

- 1. Use a separate system for development and deployment. This would cause you to have more than one Routing Table, allowing you to use a "fixed" symbolic server name. This option requires you to move the application to the Deployment server before it can be deployed.
- 2. Create two Route Tables in two separate places. Before each Version creation, use the Communications Administrator to change the location of the route table to point at the table to be shipped. For this option, someone must remember to change the Route Table path before and after each build.
- 3. Create two Symbolic Server Names (e.g. DEV and PROD). Modify the Ancestor Form to get a Registry entry that will indicate the system type, that is, DEV / PROD. To Set the Registry Value, write a function that is run with each install (after install command) to put the registry entry to indicate that this is the production system (PROD). This means that you could create one route table with both entries and have this installed on the target system. In the development area you would manually create the Registry entry with the DEV value.

Option 3 is the most flexible as it is possible to extend it further to support any number of servers.

21.3 Search Packages including an object

To check the Applications which include a specific object or group of objects use Search Application on the Tools Menu



Enter the object name, or part of the name and then select a specific application to search or search all applications.

Use the Find M button to launch the search.

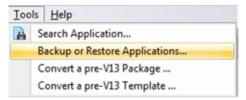
The results can be saved to a text file using the Save to File ■ option.

21.4 Back up your Application Definitions

Application information is not saved in the LANSA database but is stored in the **X_APPS** directory on your system. Applications are identified by a set of GUIDs which are generated when the first Version is successfully built. This set of GUIDs will be used to identify the Application and facilitate upgrades and patches. Without these GUIDs you cannot upgrade or patch your software product.

While this design allows flexibility to move and copy Applications easily, it does mean you need to plan to backup up this directory regularly to ensure your latest definitions, most importantly the GUIDs, and templates will not be lost in the case of a system crash.

The Backup and Restore feature is available on the Tools Menu



All Applications are saved in the X_APPS directory under the \..\ X_WIN95 \ X_LANSA folder. Nominate where this directory should be backed up to. Similarly when restoring the Application definitions nominate where Applications are to be restored from.

It is recommended that you schedule a regular back up of your X_APPS directory while you are actively using the Deployment Tool.

21.5 Deploy an external ActiveX control

Some LANSA applications will incorporate functionality provided by a 3rd-party ActiveX control. Visual LANSA supports all standard ActiveX Interfaces.

Some ActiveX controls have comprehensive installation programs to automatically register themselves on your PC (i.e. Microsoft Office ActiveX controls require Office to be installed on the target PC), however others are supplied as single .DLL or .OCX file. Shipping the single files to support an ActiveX control with your application means your distributable Package can be much smaller, however it can add complexity to the installation procedure. Alternatively you can instruct users how to manually install the ActiveX control as well as your application.

To include an ActiveX control with your deployment Package (this only works where a single file is required) you can define a Command to Execute After Install to automatically register the control for you.

The following example distributes the RICHTX32.OCX control with the deployment Package:

Step		How to do it			
	1. Create a batch file RICHTX32.BAT	Create the batch file RICHTX32.BAT:			
		Move RICHTX32.OCX %windir%\System32\			
		Regsvr32 %windir%\System32\Richtx32.OCX			
		The first line of the batch file simply moves the Richtx32.OCX to the Windows System32 directory (using standard Windows Environment Variables). The second line registers the ActiveX using			
		Regsvr32.exe			
	2. Test the batch file	Test the batch file on your own PC by placing the DLL/OCX in the same directory as the Batch file, then executing it.			
	3. Create a deployment Package	Create your deployment Package using the options relevant to your application.			
	4. Include the ActiveX component	Ensure your ActiveX component is included in the Package by directly selecting it or ensure it is included as a cross-reference object.			

5. Add non-LANSA Add the Non-LANSA objects:

objects to support your ActiveX component.

RICHTX32.OCX (this will most likely be in your

Windows\System32 directory)

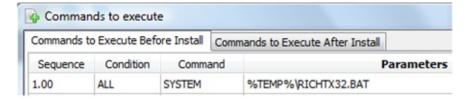
RICHTX32.BAT (which you created in step 1)

Modify the installation path for both objects to ensure

they are installed in the 'Temp directory'

6. Set up command to Select the Commands to execute and define the execute after following command to execute after installation:

installation



7. Deploy your

Save, build and release your Package then deploy as Package. required.

It is your responsibility to conform to the licensing agreements of any ActiveX control you are distributing. Some controls require specific licenses if distributed with your application.

21.6 Deploy LANSA Components published for ActiveX

When creating a LANSA application you may wish to make aspects of your application available to other development tools by publishing a selection of LANSA Components as ActiveX controls. For information on publishing a Component as an ActiveX control refer to ActiveX Wrapping in the *Visual LANSA Developer Guide*.

Deployment of these components requires that they are registered on installation and that a session configuration file is created for use when accessing these controls. These are the steps to deploy a Component, register it for ActiveX usage on the target system and automatically create the session configuration file required for it to be executed:

Step How to do it

1. Create a Create your deployment Package using the template XACTIVEX.

Package Ensure the Package options and settings are relevant to your application.

2. Add Using the Select Repository Objects dialog, select the LANSA ActiveX Component(s) published as ActiveX controls and add them to the Package.

3. Modify Open the Commands to Execute after Install.

Duplicate the comment line that has COMMAND: EXECFATAL and replace the xxxxxxxx with the compiled name for the comments.

Commands To the comment line that has COMMAND: EXECFATAL and replace the xxxxxxxx with the compiled name for the comment.

To automatically create a session configuration file in the Partition Execute directory: uncomment all the lines between the OPEN and CLOSE commands and adjust the parameters to match your installation. The configuration file contains X_RUN Arguments. The definition of each X_RUN argument can be found in the section Using the X_RUN Command in the Technical Reference.

4. Deploy Save, build and release your Package then deploy as required. your Package.

☆ It is your responsibility to deploy the non-LANSA application using the LANSA component(s) as ActiveX controls.

A LANSA license may be required to support the ActiveX control if SuperServer functionality has been embedded in the LANSA component.

21.7 Deploy User Defined RDML BIFs

To Deploy to an Execution Environment

To deploy the runtime DLL select the Built-In-Function through the Process/Function Option in the Deployment Tool.

To Deploy to a Development Environment

For Development, make sure you select the option to *Deploy to Development System* in the Setup Details (*SYSEXPORT) so that tables LX_F47 and LX_F48 are propagated to your new environment. Optionally you can select the User Defined Built-In-Function via the Process/Function to add the Built-in Function's DLL to your Package for deployment to another Development Environment.

21.8 Install Runtime License for Application

If your installed application requires runtime licenses these must be installed by executing the LANSA licensing interface.

To provide a shortcut to the licensing interface in the Application program group create an entry in the Files with Special Processing to run the process *LICENSES. For example:

When the application is installed the licensing interface can then be accessed from the Application program group



To manually execute the licensing interface use the following command:

<sysdir>\Execute> x_run proc=*licenses lang=<language>

21.9 Set DLL Version Information

When a LANSA Object is compiled specific information is generated into the DLL. To ensure your Application has the correct information in the shipped DLLs:

- 1. Move your source code to the build machine.
- 2. The Partition Security Officer Profile is needed to set the DLL Version in the following step.
 - a. Open the Compile dialog via the Visual LANSA Options > Settings > Compile dialog.
 - b. Set the DLL Version Information in the Compile options to appropriate values. The version information must be modified for each Version or Patch delivered.
- 3. Compile all compilable Application objects for the first version of the Application. For subsequent versions and patches only modified objects are required to be compiled.
- 4. Build the Version or Patch in the Deployment Tool.
- 5. Distribute MSI or MSP files to end users for installation.

For patches you MUST change the version information in order to distribute an updated DLL in the Patch. This is a Microsoft requirement as the change in version number triggers the patch builder to include the object. One issue with that is you don't know precisely what has been deployed as being able to easily see the Version number only hints at what might be in the Patch, it doesn't tell you PRECISELY what is in the Patch. You actually have to look at the patch (with a Patch Viewer) to verify the contents.

The DLL Version Information is set in the Visual LANSA IDE, like this:

Once generated into the DLL the information is visible in the DLL's property tab sheet labelled 'details' in Windows 7:

If you cannot see all the DLL Properties details you need to see, contact LANSA support for assistance.

Refer to the Compile options in the *Visual LANSA User Guide* from more information.

21.10 Specify Installation-Specific Database Connection Settings

The MSI install will always create/update the ODBC DSN specified in the install. So the installation-specific database connection settings will need to be entered during the install.

There are two ways to do this.

1. The user enters them in the MSI *Setup the Local Database* dialog or

DBSV=myservername\sqlserver DBAS=mydatabaseinstance.

2. An administrator writes a batch file or script file to pass the settings on the MSI command line. For example:

Msiexec.exe /i MYAPP_v1.0.0_en-us.msi DBII=MYAPPDSN

21.11 Deploy a 64-bit Application

If the LANSA build environment is enabled for 64-bit, any Versions and Patches will include the 64-bit runtime and selected objects during the build processing. The build will generate separate MSIs (or MSPs) for the 32-bit install and the 64-bit install.

21.12 Install Application Files into a consistent schema name

Note: This topic is only relevant for applications with files deployed to a Windows database.

When developing an application with file accessing a Windows database is it desirable to maintain the same library (schema name) through the lifecycle of the application to maintain the integrity of the database and file data. While this sounds obvious it can be problematic if not planned appropriately.

The fundamental issue relates back to RDML source version control and how various releases of the application source code are maintained, built and released.

The Problem

By default LANSA files are defined and compiled in the Default File Library of the current Partition. If the application source code is moved to a different Partition or a different LANSA system, where the partition's Default File Library is also different, the next time the file is compiled and deployed the library (schema name) will, by default, be different.

How to assign a consistent schema

- 1. BEFORE deploying the first version of the application, select a schema name (library) to be used for the installed Windows application e.g. XXXDTA.
- 2. BEFORE deploying the first version of the application, modify the application source code to include appropriate DEFINE_OVERRIDE_FILE commands to redirect the file OAMs to the appropriate LANSA Library (schema name).
 - If all file OAMs should be redirected to the selected schema XXXDTA, the code added may look something like this:

Use Builtin(Define_Override_File) (*PARTDTALIB *Default 'XXXDTA') To_

- 3. Create the first Version of the application to be deployed including the files to be created in the Windows database. Ensure the selected files are configured to install into schema name XXXDTA by creating a library substitution for this value and assigning it to the files.
- 4. When creating subsequent Versions and Patches ensure the same library substitution value is assigned to the selected files.

For more information refer to Lifecycle of an Application and Mapping a File's Library to a Fixed Deployed Schema.

Deployment Tool Tutorials

The exercises in this tutorial assume that you are an experienced Visual LANSA developer.

Exercise Structure

The exercises are divided into five groups:

Set Up the Deployment Tool must be completed before all other exercises.

Deploy Client Server Applications is a group of exercises which demonstrate how to install a client application which will connect to an IBM i server. Exercises to install Version 1, Patch 1 and Version 2 are included.

Deploy a Stand Alone Application is a group of exercise demonstrating how to install a local Windows application to the desktop. Exercises cover creating and installing Version 1, followed by a minor application change and then creating and installing Version 2.

Deploy Applications using JIT has exercises which show how to define and install a Just in Time Server, create and install a client application with JIT Update, and finally make an application change followed by creating and installing Patch 1 via the Just in Time Server.

Deploy to a Windows Server demonstrates how to create and install a Windows Data/Application Server system and then execute a client application which connects to the Windows Server application.

Except for the first exercise (DTE010 – Set Up the Deployment Tool), each group of exercises is standalone and can be completed separately.

About the Exercises

Introduction

Objects created in these exercises use a prefix if **ii**_ for object names, where **ii** are replaced by the students initials. If a unique set of initials are used both for new component names and for deployment Company and Application Name, then many students could use the same copy of Visual LANSA.

The Application to be Deployed

Each set of exercises deploy a supplied application which is copied and then deployed using the Deployment Tool. The application maintains an employee file which is part of the shipped Visual LANSA demonstration material.

Steps are included to emulate real application development, covering installing Version 1, changing the application which is then deployed via a Patch. Making a further change to the application main form, followed by creating and installing Version 2.

The application is deployed as both a client server application which may connect to an IBM i or a Windows server, and also as a standalone desktop Windows application. A deployment using a Just in Time server is also included.

Tutorial versus Real World

The exercises in this tutorial are designed to simplify deployment for training purposes and to minimize the setup involved.

Build Machine

It is essential that your own deployment uses a build machine. This is a PC with a Visual LANSA installation which is dedicated to assembling and deploying applications. As well as ensuring that all the latest and required components have been moved to this machine, the build machine approach also ensures that Product Version and File Version and GUID are all preserved and used for the next version or patch which needs to be deployed. See the *Visual LANSA Administrators Guide* for more information on the reasons for using a Windows build machine.

Deploying your own client server to an IBM i server application would involve:

• Deploy the server applications to the IBM i server via LANSA export and

import.

- Deploying the Windows client application to a file server or terminal server.
- Setup a shortcut on user desktop linking to the application top level form on the file server
- Or setting up the user desktop by running the Network Client install file provided by the file server deployment.
- Setup a Just in Time Server installation and deploy all application client software to the file server linked to the JIT server. Subsequent patches and new versions can be deployed by simply updating the JIT Server only.

Your own deployment to a Windows server would involve:

- Use the Deployment Tool to deploy the Data/Application server applications and files to a Windows server.
- Optionally create and install the client application to a file server or terminal server.
- Setup a shortcut on the user desktops linking to the client application top level form on the file server
- Or setup the user desktop by running the Network Client install file provided by the file server deployment
- Optionally create Data/Application server deployment which includes all client applications
- Set up user desktop shortcut to the top level form on the Data/Application server.
- Or setup the desktop using the Network Client install file from the Data/Application server deployment.
- Optionally set up a Just in Time server.
- Deploy both the Data/Application server applications and the client application with automatic updates via the JIT Server.

System Requirements for these Exercises

The client to an IBM i server exercises require a partition on the IBM i with the Personnel System files and OAMs from the Shipped Demonstration material save file. There are no other server functions or components required on the IBM i server.

The standalone Windows application exercises are all installed for training purposes to the student's desktop. There is no requirement for a Windows server in order to complete the exercise installations.

See the *LANSA Deployment Tool Guide /Deploy Client/Server Applications* for more information.

See also *Deploying LANSA Applications on IBM i* guide.

Set Up the Deployment Tool

This exercise is used as the first exercise in all the deployment scenarios in this tutorial.

DTE010 – Set Up the Deployment Tool

Objective

To begin using the Deployment Tool by defining a company and an application.

To achieve these objectives you must complete the following:

Step 1. Define Company

Step 2. Define Default Cross Reference Settings

Step 3. Define an Application

Summary

Before You Begin

Before you begin the exercises you should read at least the following guides and sections:

Windows Build Machine in the Visual LANSA Administrators Guide

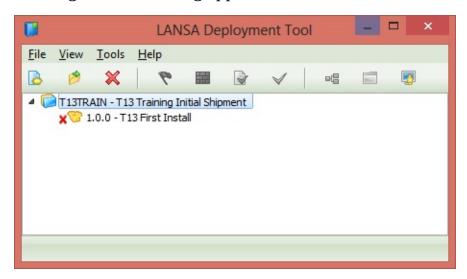
- In the LANSA Application Deployment Tool Guide:
 - Introduction
 - What's New?
 - Plan Your Deployment
 - Deploy Client Server Applications

Step 1. Define Company

1. Open the *Deployment Tool* from the *Tools* ribbon.



2. The main dialog lists all existing applications:



3. Open *Company Maintenance* from the *View* menu.



4. Click the *New* button (highlighted above). Use your initials to define a "II

Systems" company. Complete the URL entries as shown or use suitable URL's for your own employer.



Click *OK* to save your company definition.

Note: The *URL* entries are optional and may be left blank.

- 5. Before closing this dialog, open the *LANSA Application Deployment Tool Guide* from the *Help* menu and review *Set Up Company Details* in the *Configure Deployment Tool* section.
- 5. Before closing this dialog, open this guide (*LANSA Application Deployment Tool Guide*) from the *Help* menu and review *Set Up Company Details* in the *Configure Deployment Tool* section.

Step 2. Define Default Cross Reference Settings

Cross referencing allows you to select an object (for example, a Function or Component) for inclusion in a Version or Patch, and then have all, or selected types, of the related executable and definition objects automatically included at build time.

Any default options defined here can be overridden when editing the Version or Patch, but the investment of time to set up your default values correctly can bypass several repetitive steps in the definition process.

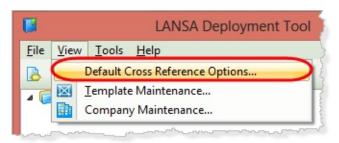
1. Before opening the *Default Cross Reference Options* dialog, open *LANSA Deployment Tool guide* and review Default Cross Reference Settings.

Note that the cross referencing facilities are extremely flexible:

- Default cross reference settings may be defined by object type.
- Defaults may always be overridden, even when options are not automatically prompted.
- *Cross Reference* is always available as objects are added to a package, including when there are no default settings defined.

Leave the online guide open, you will refer to it again in a later step.

2. Open the *Default Cross Reference Settings* dialog from the *View* menu:



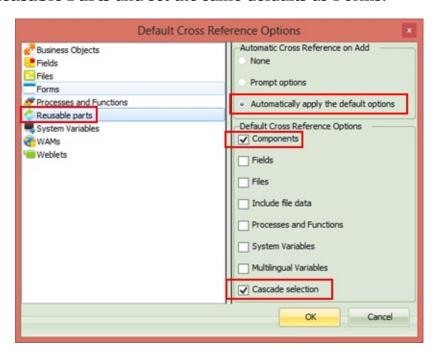
Note: When you first use the Deployment Tool, no default cross reference settings are defined for any object type.

3. Setting are defined by selecting one object type at a time and then defining the cross referencing required.

Select **Forms** and select options to:

- a. Automatically apply the default options.
- b. Cross reference Components.
- c. Select the option to Cascade Selection.

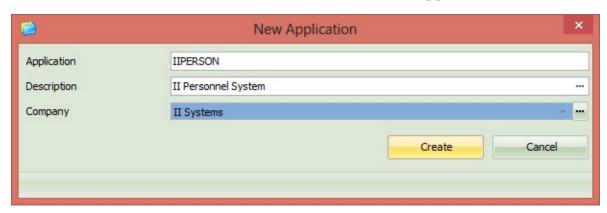
4. Select **Reusable Parts** and set the same defaults as Forms.



5. Click *OK* to save your settings and close the dialog.

Step 3. Define an Application

1. Click the *New* button on the toolbar to create an application.



2. Complete the form as shown, using your initials and selecting your company:

Application IIPERSON

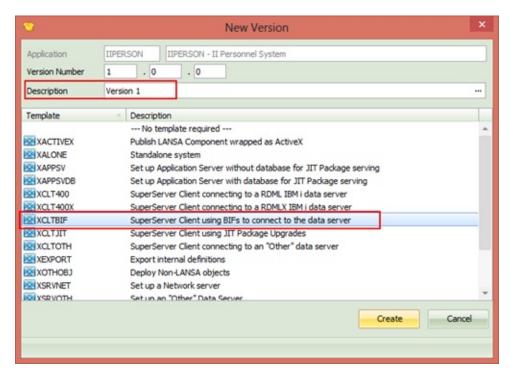
Description II Personnel System

Company II Systems

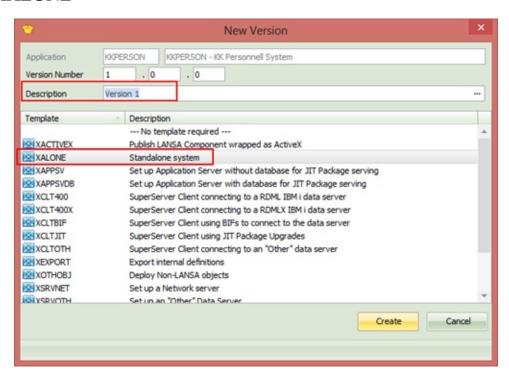
3. Click *Create* to save your application definition.

The New Version dialog will be displayed.

- 4. Note that the *Version Number* has been set to **1.0.0**. You could change these values if required.
 - a. Enter a *Description*: **Version 1**
 - b. If you plan to complete *Deploying Client Server Applications*, select template XCLTBIF.



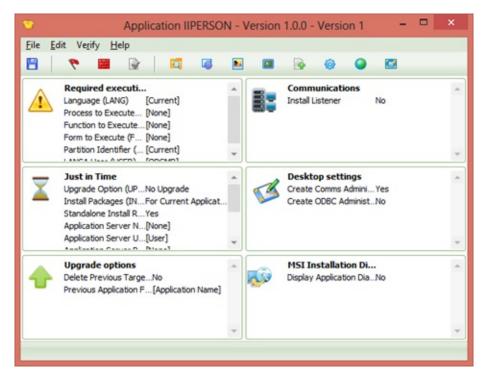
c. If you plan to complete *Deploy a Stand Alone Application*, select template XALONE



5. Review Deployment Templates in the *LANSA Application Deployment Tool* guide.

Note:

- Templates make defining a package quicker and easier and reduce complexity by hiding non required options.
- You can use the templates supplied by LANSA or create your own templates from scratch or by saving a package definition.
- Pre-V13 templates can be converted.
- 6. Click *Create* to continue. The *Package Maintenance* dialog will open:



7. Close the Package Maintenance dialog and close the Deployment Tool.

Summary

What You Should Know

- Deployment requires a company to be defined, which forms part of the installed path.
- Optionally About / Help and Update URL may be defined for company and this information is compiled into the DLL files.
- Cross Reference rules can be defined for all Repository objects, which will add related objects to the package.
- Cross Reference rule can be overridden when defining a package.
- A package should always be defined using a suitable template which make deployment easier and quicker.

Deploy Client Server Applications

This section contains exercises that demonstrate how to deploy a client application which accesses a Data/Application Server system.

The exercises to be performed to complete this objective are:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1

DTE025 – Install Version 1

DTE030 – Modify the Employees Application

DTE035 – Create a Patch for the Employees Application

DTE040 – Create Version 2

DTE015 – Create Client Server Application

Objectives

In this exercise you will create a copy of a supplied application. Later you will make small changes to this application in order to deploy a patch and then a new version. This application maintains employee records and was created using the Visual LANSA CRUD Wizard.

To achieve this objective you must:

Use Visual LANSA installed as a *Slave Workstation*, connected to an IBM i Master. The Employees Application will run as a client against the IBM i server.

Step 1. Import the supplied Employees Application

Step 2. Copy the supplied Employee Application

Step 3. Test the Client Server Application

Summary

Before You Begin

You must have completed the following exercise:

DTE010 – Set Up the Deployment Tool.

Step 1. Import the supplied Employees Application

- 1. For this step you will need Export of the Employees Application forms and reusable parts. Your instructor will tell you where to locate these export files.
- 1. Download the *Deployment Tool Employee Application from* the lansa022_extrafiles.zip file that you will find in the LANSA Documentation web page (http://www.lansa.com/support/docs/index.htm). You will find it in the Deploying LANSA group of guides. The file contains an Export of the Employees Application forms and reusable parts.

Extract the contents of the zip file into any convenient folder.

2. In Visual LANSA select the *Tools* ribbon and click on the *Import* button. Locate the folder containing the export files, select file **Ixxdir.del** and click *Open*. The *Contents to import* list should contain 2 forms and four reusable parts. Click *Import* which will populate your local repository with the following components:

II_CONNECT - Connect to IBM i

II_MAIN – Main Form

II_BRWSR - Object Browser

II_DATA – Data Object

II EDIT – Document Editor

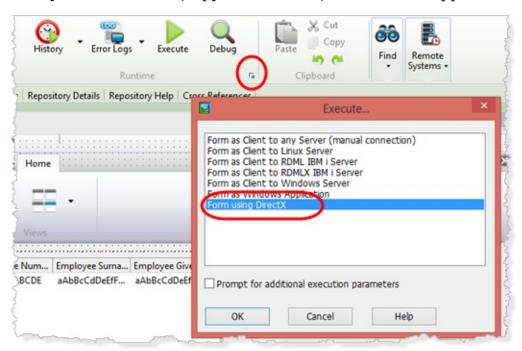
II_STORE – Data Store

Step 2. Copy the supplied Employee Application

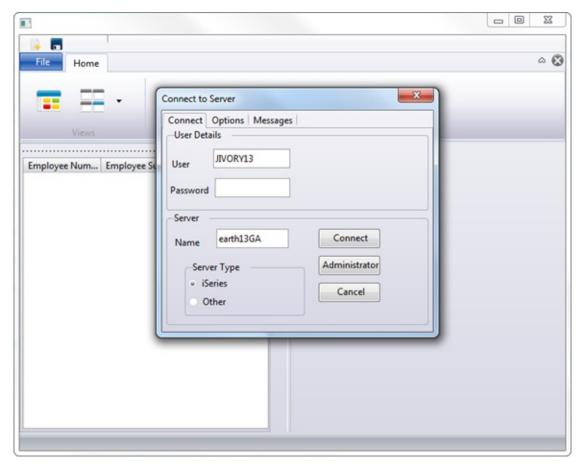
- 1. On the *Repository* tab, locate the new forms and reusable parts. Use the context menu to copy each one. Use your initials in place of II to name each component.
- 2. Select the *Source* tab for each component excluding the connect form, use *Replace* to change all occurrences of II_ to XX_ where XX is your initials.
- 3. Compile all six components.

Step 3. Test the Client Server Application

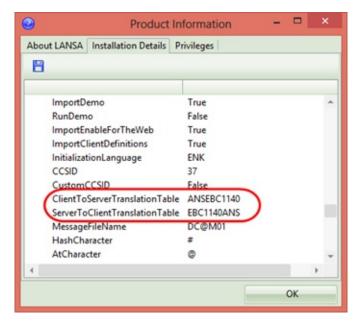
1. Execute your Main form (copy of II_MAIN) as a DirectX application.



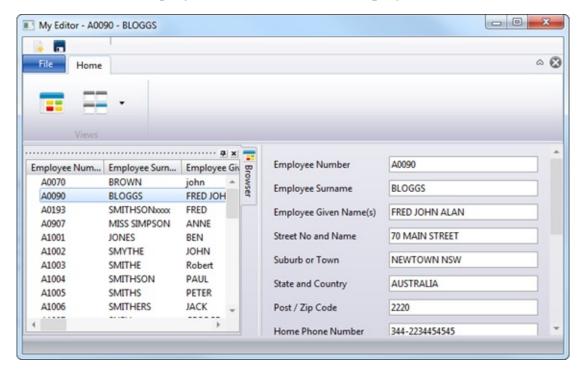
It should initially look like the following:



- 2. Complete the Connect and Options tab and click the Connect button.
- 3. Use your Visual LANSA user and password. The *Administrator* button on the *Super Server Connect* form runs the *LANSA Communications Administrator*, showing your IBM i server details (server name or IP Address).
 - You can look up conversion table names from your Visual LANSA install configuration.
- 4. Select *About* from the right hand side toolbar. In the *Product Information* dialog, select the *Installation Details* tab. Expand the InstalledSettings.cfg entry and scroll down to find the Translation Tables for your system:



- 5. Enter these values to the *Options* tab in the *Connect* form.
- 6. Once the server connection is made, the connect form signals the event uConnectionCreated. The main form then loads the DataStore (II_STORE) component which will populate itself from the Personnel file, PSLMST. The browser component (II_BRWSR) on the left of the main form is then populated with all employees.
- 7. Double click an employee in the browser to display its details:



The toolbar buttons support *Save* and *New*.

Summary

What You Should Know

• You should now be ready to start deploying the client Employees Application.

DTE020 – Create Version 1

Objective

To deploy the client application to connect to the partition you are using for training on the IBM i server. For training purposes, if you are working in partition TRN you will build a package to install to partition TRN.

If you were deploying your application into partition TST for example, then you would have promoted your application and files to partition TST. Your deployment PC would have partition TST setup in Visual LANSA and the client application would be checked out and compiled on the deployment PC. You would then build an install package (Version 1) to install the client software, probably to a file server. A Network Client install to the end user's desktop would then provide a shortcut to load and run the client application from the file server.

The target partition on the IBM i server must contain the following personnel system files:

PSLMST – Personnel

PSLSKL – Personnel Skills

DEPTAB – Department Code Table

SECTAB – Section Code Table

SKLTAB – Skill Code Table

On the IBM i, the demonstration files can be defined in the partition using *Partition Initialize* from the *Administration Menu* and selecting the save file *PERSYS – Personnel demonstration system*. The supplied files are set up to support RDMLX enabled applications.

To achieve this objective you must complete:

Step 1. Add Routing Table to Application

Step 2. Define Version 1

Step 3. Define Execution Parameters

Step 4. Review the Package Settings

Step 5. Build the Version 1

Summary

Before You Begin

You must have complete the following exercises:

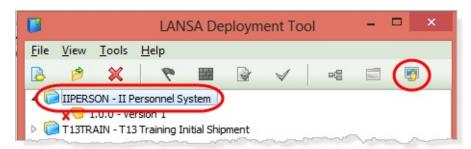
DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

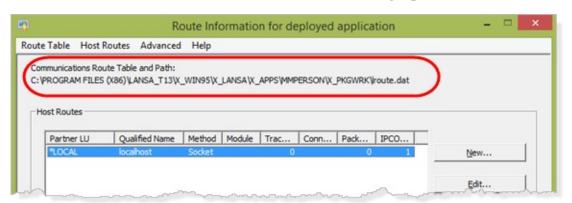
Step 1. Add Routing Table to Application

A client server application requires a routing table which LANSA communications will use, in order to connect to the server.

1. In the *Deployment Tool main window*, select the II Personnel System application and click the *LANSA Communications* button on the toolbar.



2. The LANSA Communications Administrator dialog opens:



An empty routing table (lroute.dat) is created in the package work folder (....\X_PKGWRK). You need to add a routing entry suitable for the production server with which the installed application will connect.

3. Click *New* and complete the following information to suit your IBM i server and the LANSA Listener for the LANSA system being used for training.

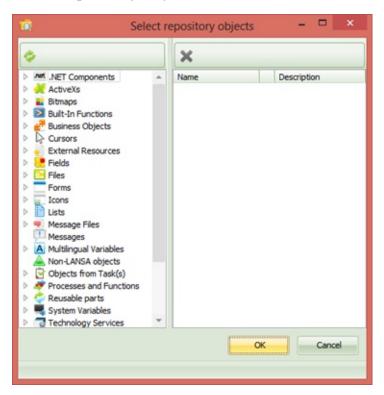
Your trainer will provide you with the information required for your workshop.

Partner LU Name: <a descriptive name for this server connection>
Fully Qualified Name of the Host: <the server name or IP Address>
Connection Identifier: <the port number used by the LANSA IBM i
Listener>

- 4. Click *OK* to save the new entry.
- 5. Close the *Route Information* dialog.

Step 2. Define Version 1

- 1. Double click on Version 1 for II Personnel System application to open the *Package Maintenance* dialog.
- 2. Click on the *Select Repository objects* 🗖 button on the toolbar.



3. Expand the *Forms* group, then the alphabetic group for your initial. Add your main form (copy of II_MAIN) to the package. Notice that the entry contains a tick meaning that cross reference is enabled for this component type.



Since the *Cascade Selection* is also selected in the cross reference defaults, it is only necessary to include this top level form to create a complete installable package. Of course depending on the design of your own applications you may need to include a number of top level components in

order to define a complete package.

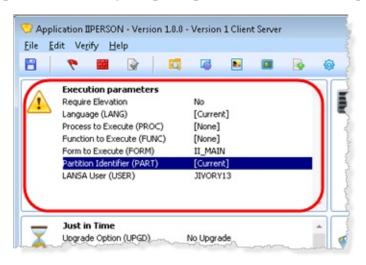
4. Click *OK* to close the *Select Repository object* dialog.

Step 3. Define Execution Parameters

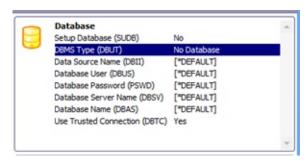
In this step you will define the execution parameters which determine the form, partition, language and user which will be used to start the application for the end user. These are parameters for the X_RUN program which loads the applications DLLs.

1. In the *Package Maintenance* dialog, define each parameter by double clicking on the parameter in the *Required Execution* group.

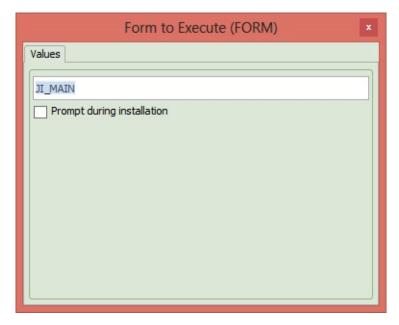
Note. These parameters may be prompted at run time if required.



2. Deselect DBMS type(DBUT) – No Database. (An error will occur if you don't do this.)



3. Double click on *Form to Execute (FORM)*:



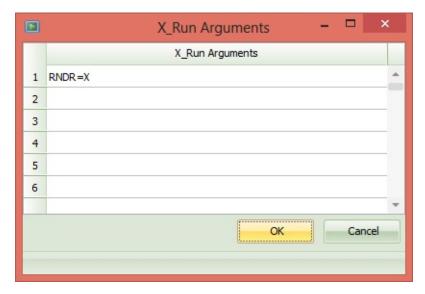
- a. Enter your form name.
- b. Close the *Form to Execute* dialog.
- 4. Follow the same procedure to define the following parameters, making any changes required for your system and national language:

LANSA User: = **<user id>**

where <user id> is your Visual LANSA User Id.

Leave all other parameters with their default settings. Note the partition and language will be values for the partition where you are building the deployment package.

- 5. The CRUD application must be executed as a **DirectX** application. This requires a change to the default *Render Type* X_RUN parameter. The *X_Run Arguments* toolbar button opens a dialog which enables these to be defined.
- 6. Open the X_ $Run\ Arguments$ dialog and add an entry RNDR=X. This will ensure the application is executed using DirectX.



7. Click *OK* to save the changes.

Step 4. Review the Package Settings

1. Click the *Settings* button on the toolbar to open the *Package Settings* dialog.



- 2. These settings are defined by the XCLTBIF template which was used to define this package. Notice that because this package is for a client application it is deployed:
 - without local database,
 - without execution database support,
 - with LANSA Communications and Component Support.

To learn more about these settings you should refer to Options & Settings.

3. Notice that the template includes a setting to *Deploy LANSA Communications*. This allows the *LANSA Communications Administrator* to be run for the installed application. This is not essential for most users. An install to a file server would enable most users to be given a shortcut to the main application form only.

4. No changes are required to these settings.

Step 5. Build the Version 1

- 1. Save the package definition.
- 2. *Check* the package. This is an optional step which will identify any errors in the current package definition without performing a build which takes a lot more time. Errors will be reported as Warning or Fatal.



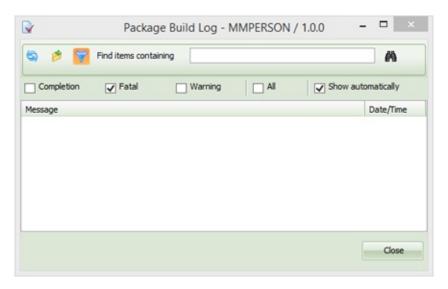
This message box is shown when no errors are found.

3. *Build* the package. A progress bar is displayed while the package build steps are completed. This step may take quite a long time.



The first time that a package is built a Global Unique Identifier (GUID) is generated for the package and stored in a file. The Install process identifies the application package in the Windows registry using the GUID.

4. When the build is completed display the build log 🔊 . Filter to select only the *Fatal* messages and you should see no messages displayed.



- 5. The build log dialog allows the messages to be filtered, for example by selecting *Completion*, *Warning*, *Fatal* or *All* messages only to be display. You can also search to *Find* specific entries in the log.
 - If you select *Completion* messages, notice that the shipped DLLs include the four reusable parts which the main form depends on, as well as the main form itself (II_MAIN) and the server connect form II_CONNECT.
- 6. Warning messages (if any) should be reviewed but generally do not require any action. For example a warning message is issued detailing the name of the GUID file generated. If fatal messages are reported you will need to review the previous steps to resolve the error. For example if a component DLL is not found, this will usually indicate the component is not compiled.
- 7. Close the *Package Build* log.
- 8. Close the Package Maintenance dialog.

Summary

What You Should Know

- How to add Repository objects to the package and how cross reference settings can be used to add dependent objects.
- How a template can define most of the settings required.
- How to define a routing entry to support connection to the server used by the client application.
- How to define additional x_run arguments
- How to build a package.
- How to review the build log.

DTE025 – Install Version 1

Objective

To install Version 1 of the II < your initials> Personnel System software.

To achieve this objective you must complete:

Step 1. Locate the Install File

Step 2. Install Version 1 of Personnel System

Step 3. Set Version 1 to Deployed

Summary

Deploying Your Own Applications

It is important to understand that these exercises will manage package definition, build and installation very differently to your own real applications. For example, your deployment package should always be built on a dedicated deployment copy of Visual LANSA. The objective is to ensure that deployment of every component is carefully controlled and the GUIDs generated during the build are preserved for building future releases of the application.

Testing on a clean deployment PC will ensure that all the required components have been identified and included in the installation.

For your own system, installation of Version 1 of your application could be to an end user desktop or to a file server or to a Terminal Server, depending on how you manage your client software.

For this exercise you will install to the partition you are using for training. The corresponding partition on the IBM i server must contain the Personnel files used by this application.

Your own deployment would normally initially be to a Test partition.

Before You Begin

You must have completed the following exercises:

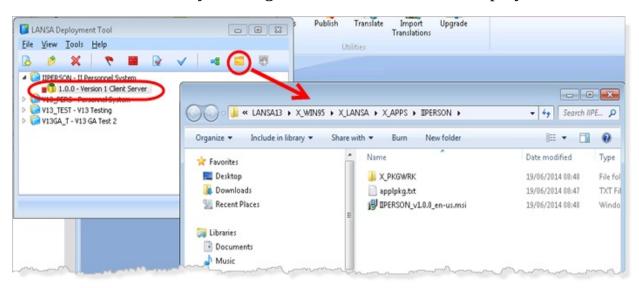
DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1

Step 1. Locate the Install File

1. Find the deployment application folder within the Visual LANSA ..\X_APPS folder by clicking on the folder icon in the Deployment Tool.



- 2. Using your initials, the deployment application folder is IIPERSON.
- 3. Using your initials, the Version 1 install file (MSI) will be:

IIPERSON_V1.0.0_en-us.msi

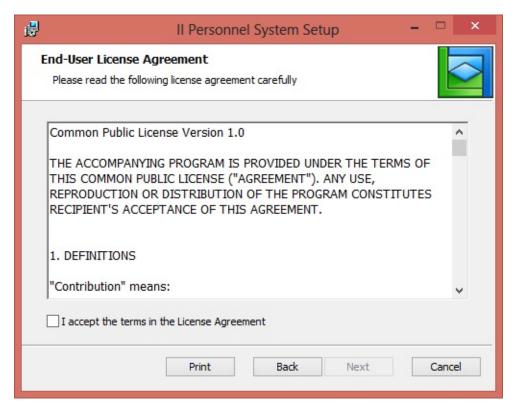
4. If you need to transfer your own install file to another machine or to any media, you only need to transfer the MSI file. No other files or folders are required.

Step 2. Install Version 1 of Personnel System

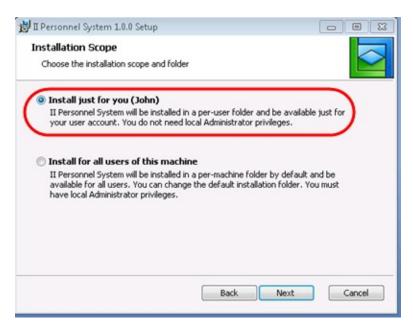
1. Double click on the MSI file to start the installation. The *Windows Install* dialog will be displayed:



- 2. Click *Next* to continue.
- 3. The *End User License Agreement* dialog is displayed.

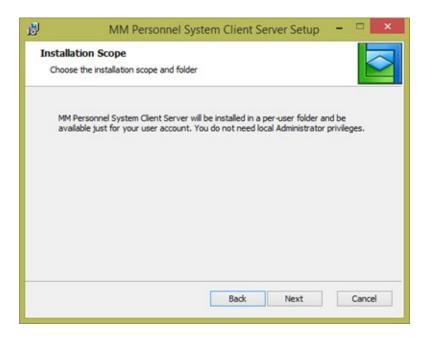


- a. A sample license agreement file is supplied in:
 - $...\x_apps\X_Wix\Content\license.rtf$
- b. You should edit or replace this file to implement your company licensing terms if necessary.
- 4. Select the "*I accept....*" check box and click *Next* to continue.
- 5. At the *Installation Scope*, select *Install just for you* and click *Next*.



If you are given a choice (as above) select *Install just for you*. This will depend on whether you are installing to Windows 7 or Windows 8 and whether UAC is enabled.

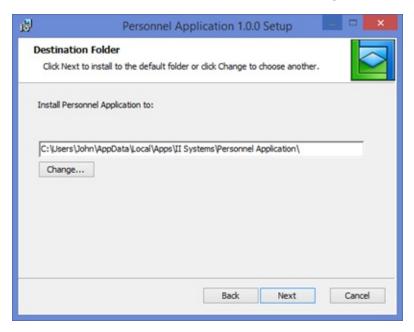
The following dialog will vary between Windows 7 and Windows 8. For example in Windows 8, since UAC is always enabled, the following dialog will be shown:



For Windows 8, you would need to install using a Command Prompt, started using *Run as Administrator*. Refer to *msiexec.exe Command-Line Options* in the *Deployment Tool Guide*.

For Windows 8, you would need to install using a Command Prompt, started using *Run as Administrator*. Refer to *msiexec.exe Command-Line Options* in this guide.

6. Click *Next* to continue. The *Destination Folder* dialog will be shown:



The *Change* button allows an alternative folder to be selected. The above image is for Windows 7 and the default path will be:

C:\Users\John\AppData\Local\Apps.

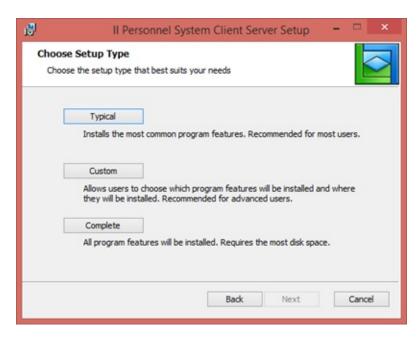
Notice that the recommended path includes the **Company** and **Application Name**:

...\II Systems\II Personnel System\

The per-user install path will be similar to the following:

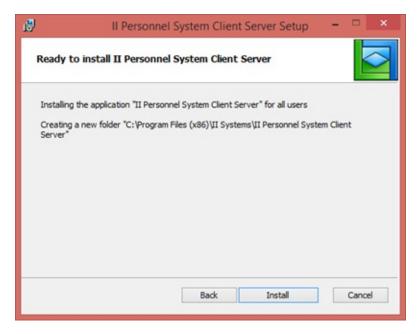
C:\Users\John\AppData\Local\Apps\II Systems\II Personnel System Standalone\

7. Click *Next* to continue. The *Choose Setup Type* dialog is displayed.

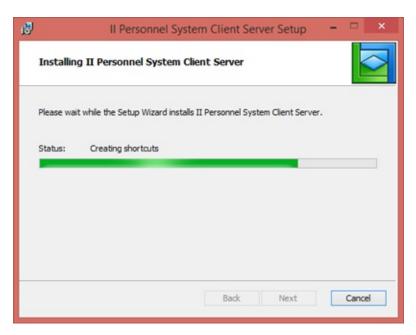


Select *any option* to continue. Currently any of these options will install the complete package.

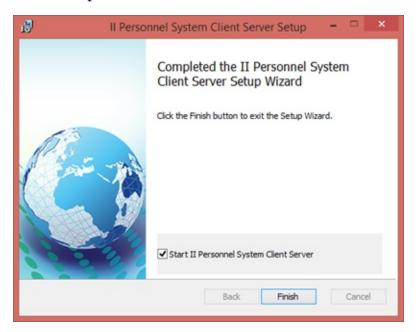
8. Click *Install* to continue.



9. The *Install* dialog will show progress with messages and a progress bar.

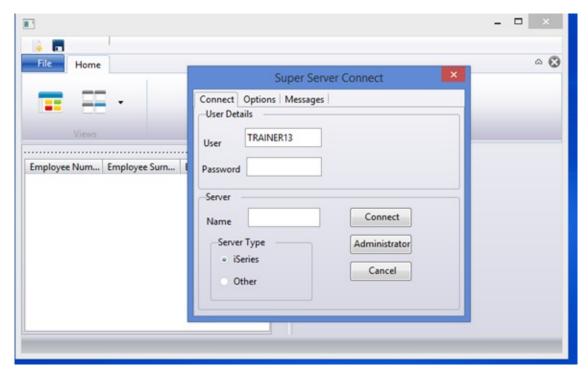


10. Click *Finish* to complete the installation.



Note that the *Start II Personnel System Client Server* check box is selected.

11. Your main form will run and invoke the Super Server Connector form.



The Server Type defaults to IBM i. Enter your User ID, Password and Server Name.

12. Select the *Options* tab which will initially show the default conversion tables. You need to enter the correct values for your system, which will depend on your IBM i code page. For example QCHRID = 37 is for a US English system. The EBCDIC to ASCII conversion tables used will depend on your national language.

Your trainer will provide the correct values for the system being used for classroom training.

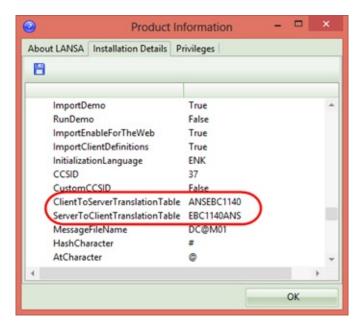
You can easily look up this information from your Visual LANSA install configuration.

To do this:

Select *About* from the right hand side toolbar.

In the *Product Information* dialog, select the *Installation Details* tab.

Expand the InstalledSettings.cfg entry and scroll down to find the *Translation Tables* for your system:



- 13. Enter these values in the *Options* tab in the *Super Server Connect* form.
- 14. Click *Connect*. After connecting with the server, the connect form will close. The main form then populates the browser panel on the left with all employees.

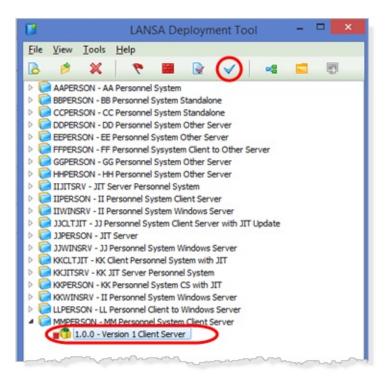
Note: The Connect form will save and restore all entries except the password.

15. As before, you are now able to edit and save an employee having selected it by double clicking in the browser or using the context menu to Open the employee details in the editor. The application handles multiple employees with each open in their own editor instance. A new employee may be created using a toolbar button. All changes are saved using a toolbar button. The application warns if closing and an employee has been changed and not saved.

Step 3. Set Version 1 to Deployed

Now that Version 1 has been successfully deployed, you should set the package status to **deployed**, which prevents any changes being made to the package.

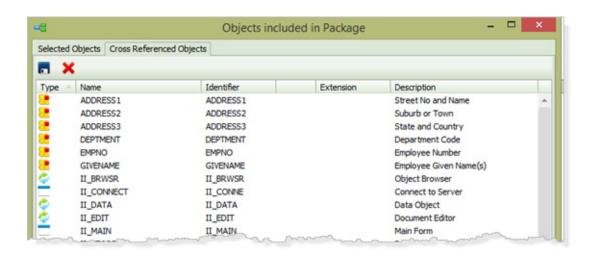
1. In the main Deployment Tool form, select the Version 1 which was just installed.



- 2. Click the Deployed button on the toolbar to set the package status to deployed.
- 3. If you now try to open the package, the following message box is displayed.



4. To check what objects were included in Version 1, use the Cross Reference button on the toolbar and select *Cross Referenced Objects* tab to see a list of all objects.



Summary

What You Should Know

- How to install using the Windows Installer.
- The default install path used for a per-user installation.
- How to set a package to Distributed.
- How to review all objects included in the distribution.

DTE030 – Modify the Employees Application

Complete this exercise if you are deploying a client server application or deploying a standalone Windows application. The changes to the Employee's Application are exactly the same for both deployments.

Objective

To make a simple change to the Employees Application which will require a Patch to be created to update the deployed application.

The real fields for start date and terminate date (STARTDTER and TERMDATER) will be removed from the application. These numeric dates have virtual fields defined over them, to transform to and from a DDMMYY (or MMDDYY) format after read and before write. The real date fields are in fact not directly used by the application.

To achieve these objectives you must complete the following:

Step 1. Remove Real Start and Terminate Dates from the Editor

Step 2. Modify the Data Component (II_DATA)

Step 3. Modify the Store component (II_STORE)

Step 4. Modify the Browser Component (II_BRWSR)

Step 5. Re-test the Employees Application

Summary

Before You Begin

You must first complete these exercises:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1

DTE025 – Install Version 1

Step 1. Remove Real Start and Terminate Dates from the Editor

In order to make the changes made visible, remove all code by making it a comment line (use **Ctrl+W**).

- 1. Using your initials, open the reusable part (II_EDIT) in the VL editor.
- 2. In the Source code use *Find* to locate STARTDTER. The code for TERMDATER is at the same location.
 - a. Comment out the following 4 lines of code to remove the fields from the user interface.

Define_Com Class(#STARTDTER.Visual) Name(#STARTDTER) Componentversion(1) Displayposition(10) Height(21) Left(15)

Parent(#COM_OWNER) Tabposition(10) Top(294) Usepicklist(False) Width(493)

Define_Com Class(#PRIM_ATLI) Name(#ATLI_STARTDTER)

Attachment(Top) Manage(#STARTDTER) Marginbottom(5) Marginleft(5)

Marginright(5) Margintop(5) Parent(#ATLM_Editor)

Define_Com Class(#TERMDATER.Visual) Name(#TERMDATER)

Componentversion(1) Displayposition(12) Height(21) Left(15)

Parent(#COM_OWNER) Tabposition(12) Top(356) Usepicklist(False) Width(493)

Define_Com Class(#PRIM_ATLI) Name(#ATLI_TERMDATER)

Attachment(Top) Manage(#TERMDATER) Marginbottom(5) Marginleft(5)

Marginright(5) Margintop(5) Parent(#ATLM_Editor)

3. Comment the following 2 lines from the component **Initialize** event routine.

#STARTDTER := #Document.p_STARTDTER #TERMDATER := #Document.p_TERMDATER

4. Comment out these two event handling routines:

Evtroutine Handling(#STARTDTER.Changed)

#Document.p_STARTDTER := #STARTDTER

#EditorModified := True

Signal Event(Changed)

Endroutine

Evtroutine Handling(#TERMDATER.Changed)

#Document.p_TERMDATER := #TERMDATER

#EditorModified := True Signal Event(Changed) Endroutine

5. Compile II_EDIT

Step 2. Modify the Data Component (II_DATA)

- 1. Open your version of the II_DATA reusable part in the editor.
- 2. Comment out these two lines:

Define_Pty Name(p_STARTDTER) Get(*AUTO #STARTDTER) Set(*AUTO #STARTDTER) Define_Pty Name(p_TERMDATER) Get(*AUTO #TERMDATER) Set(*AUTO #TERMDATER)

3. Recompile the II_DATA component.

Step 3. Modify the Store component (II_STORE)

- 1. Open your version of the II_STORE reusable part in the editor.
- 2. Comment out the following two lines:

```
#Document.p_STARTDTER := #STARTDTER
#Document.p_TERMDATER := #TERMDATER
```

3. Comment out the following two lines from the **PopulateProxy** method routine:

```
#Proxy.p_STARTDTER := #STARTDTER
#Proxy.p_TERMDATER := #TERMDATER
```

3. Comment out the following two lines from the **ExtractDocument** method routine:

```
#STARTDTER := #Document.p_STARTDTER
#TERMDATER := #Document.p_TERMDATER
```

4. Recompile the II_STORE component.

Step 4. Modify the Browser Component (II_BRWSR)

- 1. Open your version of the II_BRWSR reusable part in the editor.
- 2. Comment out the following two lines which define the tree view columns:

Define_Com Class(#PRIM_TVCL) Name(#TVCL_STARTDTER)
Displayposition(10) Level(2) Parent(#Tree) Sortonclick(True)
Source(#STARTDTER) Width(30) Widthtype(Fixed)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_TERMDATER)
Displayposition(11) Level(2) Parent(#Tree) Sortonclick(True)
Source(#TERMDATER) Width(30) Widthtype(Fixed)

3. Comment out the following two lines from the **AssignColumnData** method routine:

```
#STARTDTER := #Proxy.p_STARTDTER
#TERMDATER := #Proxy.p_TERMDATER
```

4. Recompile the II_BRWSR component.

Step 5. Re-test the Employees Application

- 1. Execute your main form.
- 2. Confirm that the application runs correctly without errors.
- 3. Confirm that the real start date and terminate date (YYMMDD) are not shown in the browser or the details panel.
- 4. Change and update an employee.
- 5. Create an employee and save it.

Summary

What You Should Know

• Which components were changed and what changed on the user interface (browser and editor).

DTE035 - Create a Patch for the Employees Application

Complete this exercise if you are deploying a client server application or deploying a standalone Windows application. The steps to follow to create Patch 1 are exactly the same for both of these deployments.

Objective

To create and install Patch 1 which includes the changed reusable parts created in DTE 0.30.

To achieve this objective, you should complete the following:

Step 1. Set DLL Version and Recompile Changed Components

Step 2. Create a Patch

Step 3. Install the Patch

Summary

What is a Patch?

A Patch is used to deliver a set of changes to a software product that has been installed using the Windows Installer. A software product can be upgraded by installing a new Version (MSI file) or by applying a Patch (MSP file).

A Patch is typically used to apply modified compiled objects, shortcuts and executables. It is not recommended to use a patch to deploy database changes as they are complex to manage and can easily result in a corrupted database.

A Patch is identified by its Patch Number. The Patch number has a direct relationship to the Version or Patch it was based on, for example Patch 1.0.0.1 would be the first patch against Version 1.0.0.0.

If a patch is only to be used to ship updated objects and there are no new objects, then nothing needs to be changed in the Package Definition. Just save the Package and build it. Any differences between the Version and the current state of objects will be included in the Patch.

A Patch is usually created from the last Patch definition, so that it includes any changes made to the Package Definition, like the addition of new objects.

The order of the application of Patches does not change the installed result. If Patch 1 contains MYFORM.DLL version 1.0.0.1 and Patch 2 contains MYFORM.DLL version 1.0.0.2 then whether Patch 1 is installed first or Patch 2 the result is that MYFORM.DLL version 1.0.0.2 is installed.

Similarly when a Patch is uninstalled. If Patch 1 is now uninstalled, it will only affect the version of DLLs which it installed. As version 1.0.0.2 is installed, uninstalling Patch 1 will not replace MYFORM.DLL with version 1.0.0.0. It will still be version 1.0.0.2. Only when Patch 2 is then uninstalled will it be restored to version 1.0.0.0.

This may sound neat, but it can become complex to manage if Patches are created which are not cumulative. Each Patch should include all the changes that have come before, or back them out if necessary, and be considered as the ONLY patch required to move the installation to that Patch level. So no matter what Patch level a user has chosen to install, to get to the latest version they just need to install the latest Patch. No interim Patches are required. Note: By creating patches sequentially from a single build machine and not restoring any objects to a prior state will ensure that Patches are cumulative. See the *Deployment Tool* guide for further information on patches.

Creating a Patch

The process to create a patch is very simple. The patch is based on the last deployed version. A patch may include one or more changes to existing application components that have already been deployed, or new components.

You modify one or more components and then build a patch based on the latest version deployed for this application. The deployment tool will analyze the DLLs and include only new or changed DLLs in the patch.

Before You Begin

You must have completed the following exercises:

DTE015 – Create Client Server Application

DTE020 – Create Version 1

DTE025 – Install Version 1

DTE030 – Modify the Employees Application

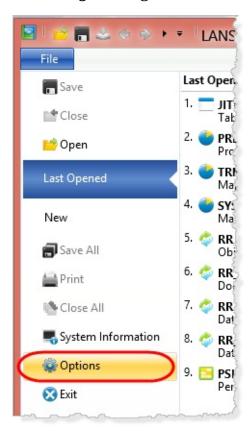
Step 1. Set DLL Version and Recompile Changed Components

Note: This step is necessary so that when Patch 1 is built, the patch recognizes the changed DLLs and only these are included in the Patch.

To complete this step you must log in to Visual LANSA as the Partition Security Officer.

If you are using an Independent Visual LANSA installation, the profile PCXUSER is Partition Security Officer.

1. Open the *Visual LANSA Settings* dialog from the *File* menu:



2. Select the *Compile* icon. Your Version 1 install used a product version number of 1.0.0, which is the default value. As shown, set a File Version which is one higher than the current product version, that is 1.0.0.1 in this case.

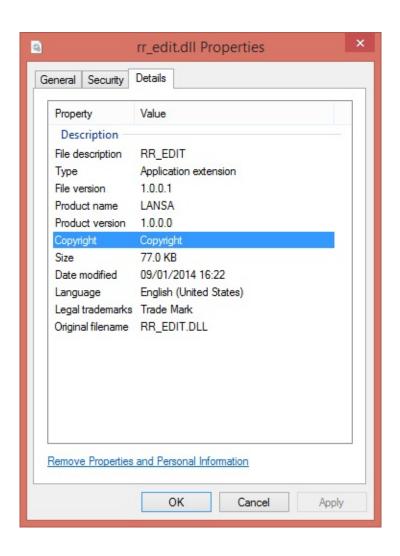
Enter any suitable value for Copyright, Trade Mark and Comment.



For your own deployment packages, you should manage *Product Version* and *File Version* as necessary for each application that you are deploying, depending on whether this is a version or a patch. This information is then built into every DLL.

For more information, refer to Set DLL Version Information.

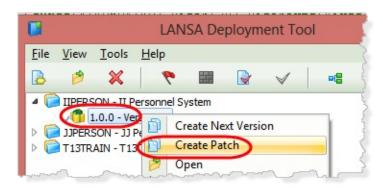
- 3. Click *Apply* and then *OK* to close the *LANSA Settings* dialog.
- 4. On the *Favorites / Last Opened* tab, select the four reusable parts which you changed in DTE030 Modify the Employees Application and recompile them. The DLLs now contain the new *File Version* information.
- 5. To view the information on any DLL, select it, right click and select *Properties* from the context menu. Select the *Details* tab:



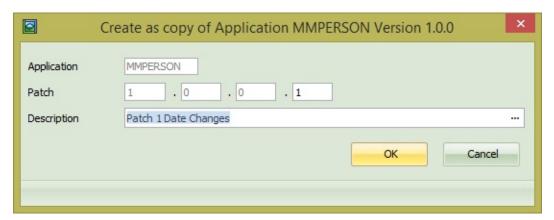
Step 2. Create a Patch

Now that you have recompiled your changed components with the new *File Version* information, in order to create a patch there is very little to do. The patch is based on the previous version, and when built, an MSP file will be produced containing only the new or changed DLLs which were part of the previous version.

1. Open the Deployment Tool. In the main window, select the Version 1 package in the Personnel Application. Use the context menu and select *Create Patch*:



2. Notice that the *Create Patch* dialog is headed *Create as a copy of Application IIPERSON Version 1.0.0*. When you build the Patch package the differences between current application components and the last deployed version are analyzed and a patch install file (MSP) is created containing only the changed components.



Accept the Patch number generated (1.0.0.1).

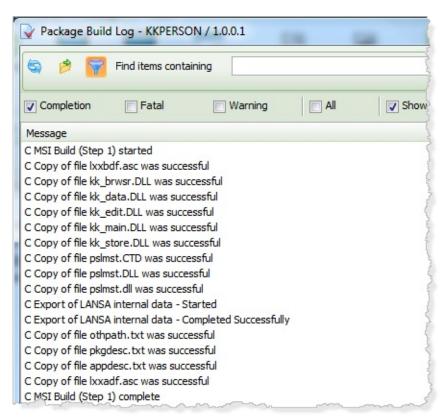
- 3. Click *OK*.
- 4. The *Package Maintenance* dialog will open. Note that most of the options

are disabled. These will be as defined in the Version 1 package.

5. Your application changes have created a new version of four reusable parts: II_EDIT, II_STORE, II_DATA and II_BRWSR.

When you build the patch package, the deployment tool will analyze all the components included in Version 1 and create a patch package install file (MSP) containing only the changes, in this case the four DLLs list above.

- 6. In the Package Maintenance dialog, click Save.
- 7. *Check* the *Package*. No errors should be found.
- 8. *Build* the Package. A message box is displayed prompting a rebuild of the cross references. In this case answer **No**, as there have been no changes to the structure of the application. You would need to rebuild the cross references if your application changes had introduced new components used by the main form II_MAIN.
- 9. Review the build log. You should have no fatal or warning messages If you filter for *Completion* messages, and scroll through the build log to find the copy DLLs steps, you will find all the application DLLs are listed at this stage.



Note: If you examine the contents of the patch install file (MSP) you will find it contains only the changed DLLs. In this case it will contain only the four changed DLLs.

If you have a utility such as 7-Zip installed (open source software) you can view the contents of the MSI or MSP file.

10. Close the Package Maintenance dialog.

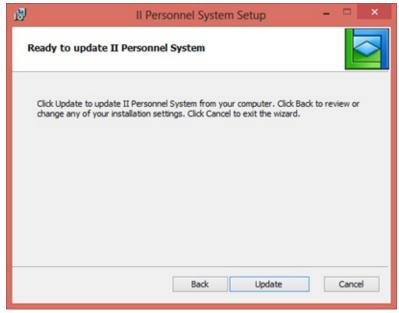
Step 3. Install the Patch

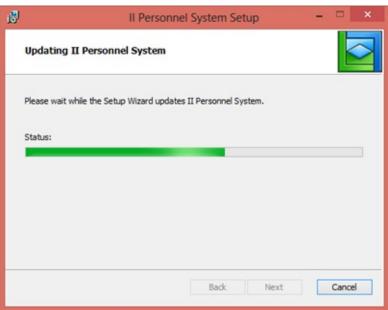
In this step you will follow essentially the same steps as when you installed Version 1 of the Personnel System.

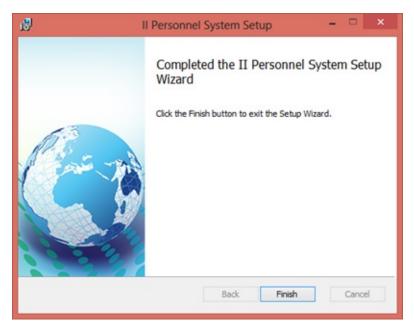
- 1. Locate the install file by selecting the patch in the *Deployment Tool* and click on the *Go to folder* □ toolbar button to open Explorer at the ... \X_APPS\IIPERSON folder.
- 2. Double click on the patch file. e.g. IIPERSON_V1.0.0.1_en-us.msp

 Note: This patch installs four changed program files (DLLs) and involves no database changes. The application was installed by Version 1 as a per-user installation. This means that in Windows 7 or Windows 8 the patch can be installed by double clicking the MSP file.
- 3. The Windows Installer will locate the installed software and confirm that Patch 1 is for the GUID as Version 1.
- 4. There are fewer dialogs to display during a patch installation:









- 5. Click the *Finish* button to close the install program.
- 6. Run the Employees Application. Notice that the employee details panel now includes only the virtual fields for start date and terminate date in DDMMYY (or MMDDYY) format. The start date and terminate date in YYMMDD format are no longer shown on the details panel and they are not included as columns in the browse panel.

Summary

What You Should Know

- How to set the Compile option settings to increment Product Version / File Version.
- How to create a patch based on the previous Version.
- How to install a Patch.

DTE040 – Create Version 2

Objective

To make a minor change to the appearance of the application and then create Version 2 to install the changed application.

What is a New Version?

A new version is created by copying the original version. The new version will contain all the components installed by the original version. The Windows Installer will process a Version upgrade as a full installation. If an earlier version is located on the target system the Windows Installer will un-install the older version before installing the new version.

Version or Patch Upgrade?

The main advantage of a Patch is it is much smaller and therefore more easily (more quickly) installed. A Patch only install changes. A Patch must detect an install with the same GUID. See the *Deployment Guide* for more information.

To achieve this objective, complete the following:

Step 1. Set DLL Version

Step 2. Enhance the Application

Step 3. Create Version 2 Package

Step 4. Install Version 2

Summary

Before You Begin

You must have completed the following exercises:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1

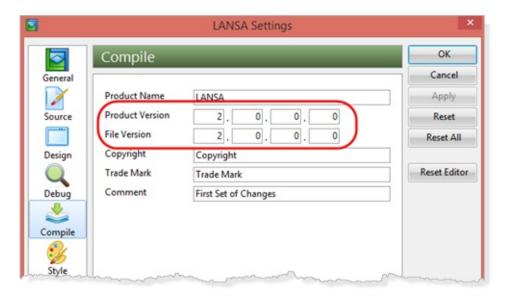
DTE025 – Install Version 1

DTE030 – Modify the Employees Application

DTE035 – Create a Patch for the Employees Application

Step 1. Set DLL Version

- 1. Open the Visual LANSA Settings dialog by selecting *Options* from the *File* menu.
- 2. Select *Compile* options. Set both the **Product Version** and **File Version** to 2.0.0.0



The *Product Version* and / or the *File Version* must be incremented before compiling the changed application. The following values would also be valid, for example:

Product Version: 2.0.0.0

File Version: 1.0.0.0

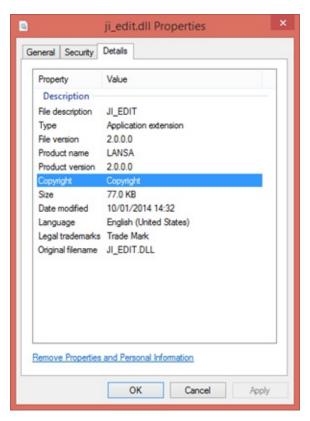
3. Click *Apply* and then *OK* to close the *Settings* dialog.

Step 2. Enhance the Application

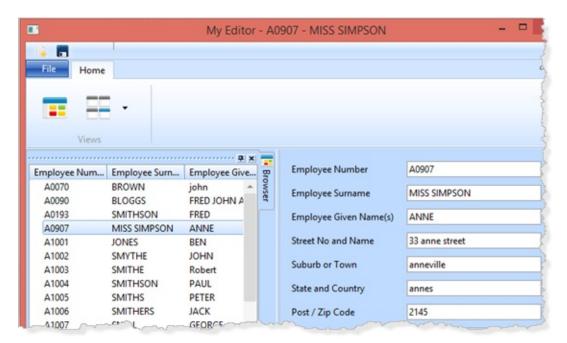
Since the main focus in these exercises is to learn how to use the Deployment Tool, the changes made in this step will be trivial. The changes will make it visually obvious that you're running Version 2 of the Application.

The main form will now use the theme 2007Blue (instead of 2007Silver).

- 1. Open your main form (copy of II_MAIN) in the editor. Locate the following code, and change it as shown. Changes are highlighted in red.
 - * Determine our appearance. #SYS_APPLN.ThemedForms := True #SYS_APPLN.Theme := 2007Blue
- 2. Recompile the form.
- 3. On the *Favorites / Last Opened* tab select your reusable parts (copies of II_EDIT, II_DATA, II_STORE and II_BRWSE) and compile them.
- 4. Select your connect form (copy of II_CONNECT) and recompile it.
- 5. All the application components now have the new Product Version and File Version compiled into their DLLs



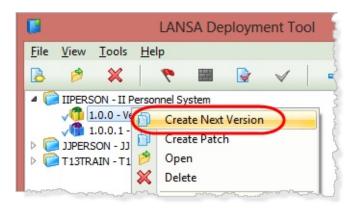
6. Run your main form and confirm that the application now uses the 2007Blue Windows theme:



Step 3. Create Version 2 Package

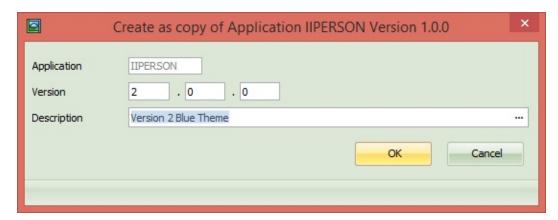
This step creates Version 2 for the Personnel System application. A version always deploys the complete application and replaces the existing installed application.

1. Open the Deployment Tool. Expand the Personnel System, select Version 1 and use the context menu to *Create Next Version*:



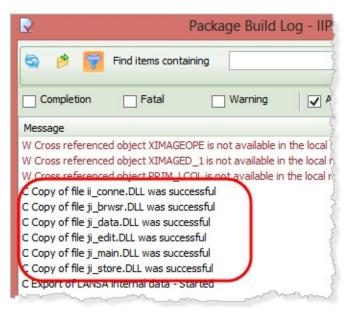
2. Enter *Version Number*: **2.0.0** and a *Description*: **Version 2 with Blue Theme**.

The *Package Maintenance* dialog will open.

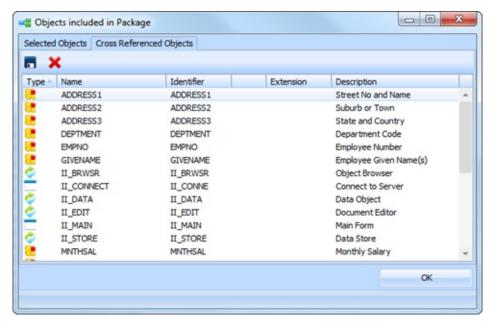


- 3. Open the *Select repository objects* dialog, select Forms and confirm that as per Version 1, the application objects are defined by the main form (copy of II_MAIN) and cross reference drill down, which includes the reusable parts (copies of II_EDIT, II_STORE, II_DATA and II_BRWSE and the form II_CONNECT).
- 4. Click *Save* and then click *OK* to close the *Package Maintenance* dialog and *Save* the package.

- 5. Select Version 2 and use the toolbar buttons to *Check* the Package and then *Build* the Package. Click *No* at the message box "Do you want to rebuild list of cross referenced objects". In this case this is unnecessary. But consider the situation where the enhancements made had included adding a new form or reusable parts. In this case it would be essential to regenerate the cross reference so that all objects are included.
- 6. When the build is complete, check the build log, filtering for *Completion* messages. Scroll down to the copy DLLs step. Confirm that all your components have been included:



- 7. Close the build log.
- 8. In the Deployment Tool main window, select Version 2 for the Personnel System application. Click the *Objects Included in the Package* tool bar button.

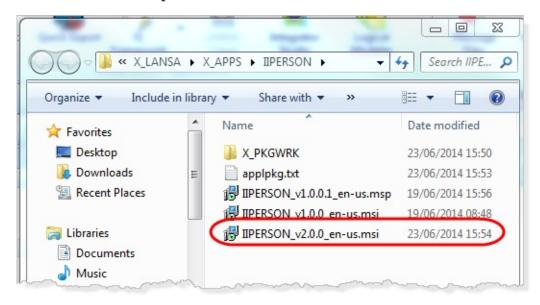


This form contains two tab sheets.

- *Selected Objects* contains the components which were defined by the objects you included in the *Package Maintenance / Select repository objects* dialog.
- The *Cross Reference Objects* lists the objects included by cross reference and cross reference drill down.
- The *Delete* option and can be used to remove objects from the package. If any objects are removed from the package, the Package will need to be rebuilt.
- If any objects added to the package by cross reference are removed, rebuilding the package will regenerate the cross referenced objects if the option to "regenerate cross reference" is taken.
- 9. Close the Objects included in the Package dialog.

Step 4. Install Version 2

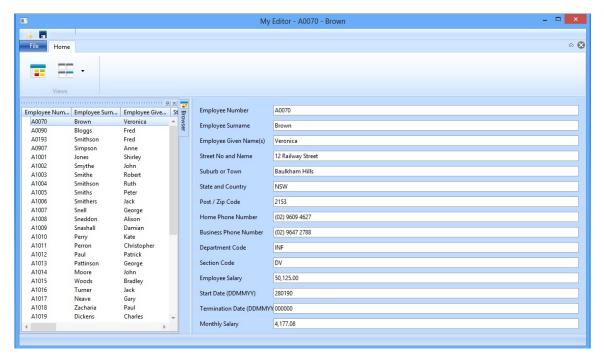
1. Select Version 2 and use the *Go to folder* toolbar button to open Explorer at the application package folder (..\x_apps\IIPERSON) in your Visual LANSA install path:



2. Double click on the Version 2 MSI file to run the installation.

Note: Version 2 installs new versions of all forms and reusable parts. Once again it does not involve any database changes, and Version 1 was installed as a per-user installation. This means that for both Windows 7 and Windows 8 you can install Version 2 by double clicking the MSI file.

- 3. You will observer that installing a new version is similar to the initial install. For example, the *End User Licence Agreement* is displayed and must be accepted.
- 4. Complete all the install steps for Version 2. Version 2 will install as a peruser installation, the same as the option used when Version 1 was installed.
- 5. Run the application and notice that the main form and the connect form now use the 2007 Blue theme:



6. Log in to the server and re-test the application. You should find that the only change is to use the 2007 Blue theme.

Summary

What You Should Now Know

- How to create a Version 2 package, based on Version 1.
- How to install a Version 2 package.

Deploy a Stand Alone Application

This section contains a series of exercises to demonstrate how to deploy a standalone desktop application.

The exercises to complete this deployment example are:

DTE010 – Set Up the Deployment Tool

DTE045 – Create a Stand Alone Application

DTE050 – Create Version 1

DTE055 – Install Version 1

DTE060 – Modify the Employees Application

DTE030 – Modify the Employees Application

DTE035 – Create a Patch for the Employees Application

DTE070 – Create Version 2

DTE040 – Create Version 2. This exercise includes installing Version 2. Ignore reference to the form II_CONNECT which is not used by the stand alone installation.

DTE045 - Create a Stand Alone Application

Objective

In this exercise you will copy a supplied stand alone Employee Application. It will run locally on the desktop using a local SQL Server database. Complete this exercise if you do not have access to an IBM i server which would enable a client/server application to be deployed, or if you simply want to deploy a desktop application.

Later you will make small changes to this application in order to deploy a patch and then a new version.

To achieve this objective you must complete the following:

Step 1. Import the supplied Employee Application

Step 2. Copy the supplied Employee Application

Step 3. Test the Stand Alone Employees Application

Summary

Once you have completed this exercise, continue with DTE050 – Create Version 1.

Before You Begin

You must have completed this exercises:

DTE010 – Set Up the Deployment Tool

Step 1. Import the supplied Employee Application

- 1. Download the Deployment Tool Stand Alone Employee Application (lansa022_extrafiles.zip) file from the LANSA documentation web page (http://www.lansa.com/support/docs/index.htm).
- 2. Extract the zip file into any convenient folder. The file contains an export of the Stand Alone Employee Application forms and reusable parts.
- 3. In Visual LANSA select the *Tools* ribbon and click on the *Import* button. Locate the folder containing the export files, select the file **lxxdir.del** and click *Open*. The *Contents to import* dialog should contain 1 form and 4 reusable parts. Click *Import* to populate your local repository with the following components:

JI_MAIN – Main Form

JI_BRWSR - Object Browser

JI_DATA – Data Object

JI_EDIT – Document Editor

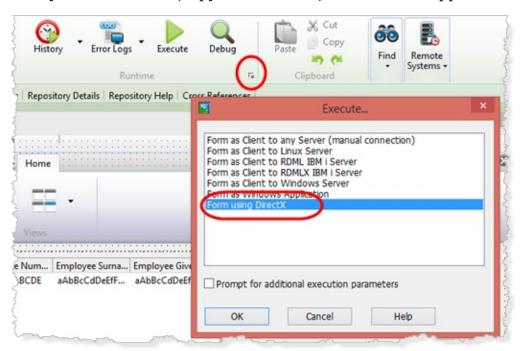
JI_STORE – Data Store

Step 2. Copy the supplied Employee Application

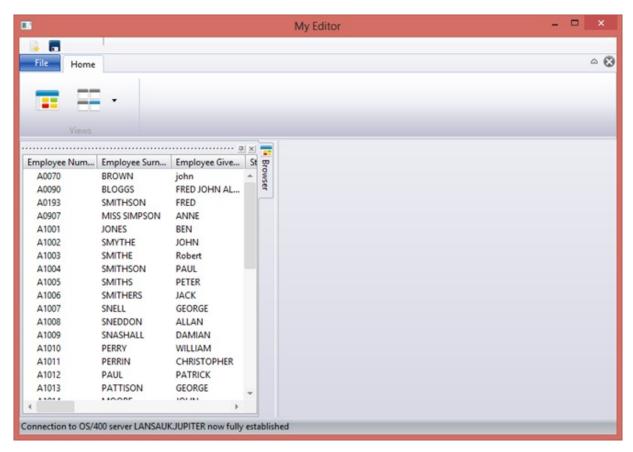
- 1. On the *Repository* tab, locate the new forms and reusable parts. Use the context menu to copy each one. Use your initials in place of **JI** to name each component.
- 2. Use *Replace* to change all occurrences of JI_ to XX_ where XX is your initials, for:
 - a. each component excluding the connect form
 - b. the source code.
- 3. Compile all five components.

Step 3. Test the Stand Alone Employees Application

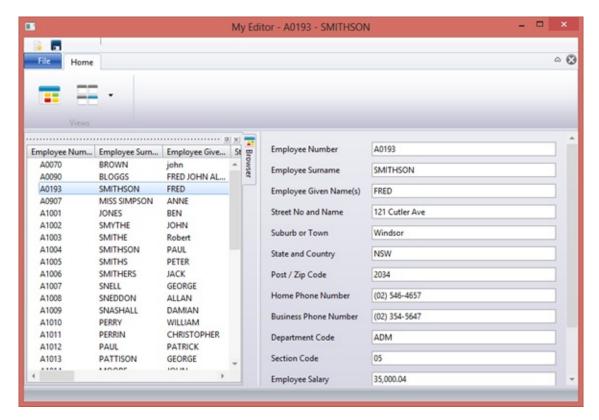
1. Execute your main form (copy of JI_MAIN) as a DirectX application.



It should initially look like the following:



- 2. The application initially loads all employee records into the browser component, displayed on the left hand side.
- 3. Double click an employee in the browser to display its details. Multiple employees may be opened, each one in its own editor instance.



The toolbar supports *Save* and *New*.

4. The employee browser panel may be floated. Dock it back on the left hand side by double clicking its *Title* bar.

If the browser panel has been closed it may be re-opened by clicking the browser toolbar button.

Summary

What You Should Know

• You should now be familiar with the application you will use to deploy and then modify and deploy the changes.

DTE050 – Create Version 1

Objective

To create an initial installation for a standalone Windows application. The package will be defined using the XALONE template.

To achieve this objective you will complete the following:

Step 1. Create an SQL Server Database

Step 2. Define Version 1

Step 3. Define Execution Parameters

Step 4. Define the Database Parameters

Step 5. Review the Package Settings

Step 6. Build Version 1

Summary

Deploying Your Own Applications

When initially deploying your own applications you will want to deploy to a test partition.

For training purposes you will be deploying the application to a new SQL database into the partition which you are using for training. If your training partition is TRN, you will deploy to TRN.

Since this is a deployment to a new database, as well as the application, you will include the files used by the application and the file data. If you were deploying another application to this database using the same files, the files would not need to be included. If you later needed to deploy changes to any files, you would do this by creating a new version for the application which originally deployed the files.

In this application, the primary file is PSLMST – Personnel. When you deploy LANSA defined files, it is important to remember that validation rules and predetermined join fields will refer to other files. For example, the Department Code field in the file PSLMST must exist in the lookup table DEPTAB. Your deployment must therefore include file PSLMST and also the files it depends on.

When you are managing your own application deployment, if you develop in partition DEV and deploy to partition TST, your setup should include:

a. Developer PC with Visual LANSA using partition DEV

- b. A Deployment PC with Visual LANSA using partition TST. Your application definition is checked out to this VL system via an IBM i Master or a Master VCS system
- c. Deployed applications located on a Windows server, a Terminal Server or directly installed on an end user desktop.

Before You Begin

You must have completed the following:

DTE010 – Set Up the Deployment Tool

DTE045 – Create a Stand Alone Application

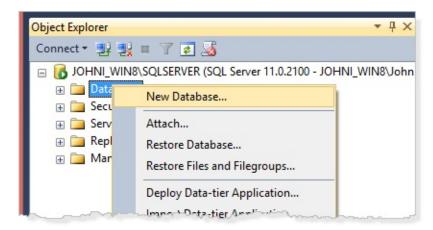
Step 1. Create an SQL Server Database

You must have *SQL Server Management Studio* installed to complete this step.

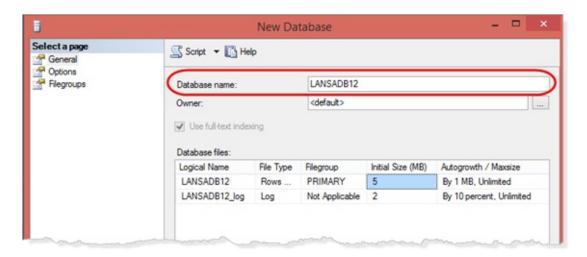
1. Start *SQL Server Management Studio*. When prompted connect to the local database server:



2. Right click on *Databases* in the *Object Explorer* tree view and select *New Database*:



3. Enter a new database name and click *OK*



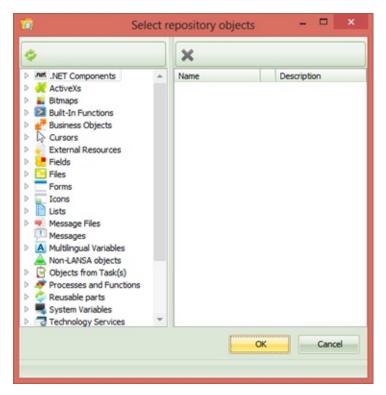
4. You have created a new SQL Server database using default settings. You will now create a deployment package to install the standalone Windows application which connects to this database.

Step 2. Define Version 1

You defined your Personnel System application and Version 1 in the deployment tool in exercise DTE010 – Set Up the Deployment Tool. You will now complete the definition of this Version 1 package and then build the package.

Note: In DTE010 – Set Up the Deployment Tool you must have created your Version 1 package using the **XALONE** template.

- 1. Open the *Deployment Tool* from the *Tools* ribbon. Double click on the *Version 1* package for the **Personnel System** application, to open the *Package Maintenance* dialog.
- 2. Click on the *Select repository objects* 📮 button on the toolbar.

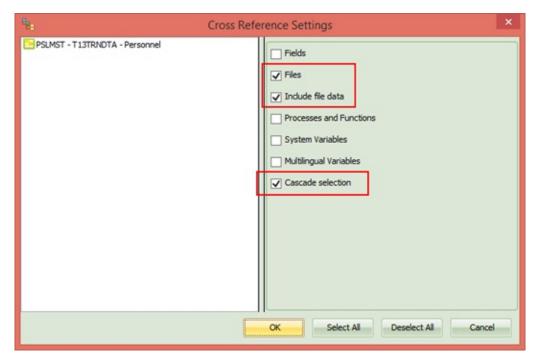


3. Expand the *Forms* group, then the alphabetic group for your initial. Add your copy of the JI_MAIN form to the package. Notice that the entry shows a tick, meaning that cross reference is enabled for this component type.



Since the *Cascade Selection* option is selected in the cross reference defaults, it is only necessary to include this top level form to create a complete installable package. Of course depending on the design of your own applications, you may need to include a number of top level components in order to define a complete application package.

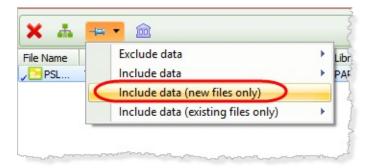
- 4. Expand the *Files* group and then files beginning **P**. Add file PSLMST to the package.
- 5. Select the file PSLMST and click the *Cross Reference* tool bar button.
- 6. Select the *Files* option, so that files which the employee file depends upon are included in the package. For example file look up rules or predetermined join fields will mean that the OAM for file PSLMST accesses other files such as DEPTAB and PSLSKL.



7. Also select *Include file data* and *Cascade selection*. This means that file data

will be included for the additional files which are included due to cross referencing. The *Cascade selection* option will mean that other files which the cross referenced files depend on, will be included.

- 8. Click *OK* to save this setting.
- 9. Notice that another toolbar button, the *Include File Data* button, enables files to be deployed including their data.



This exercise will deploy to a new database. This means it is essential to deploy the files with data.

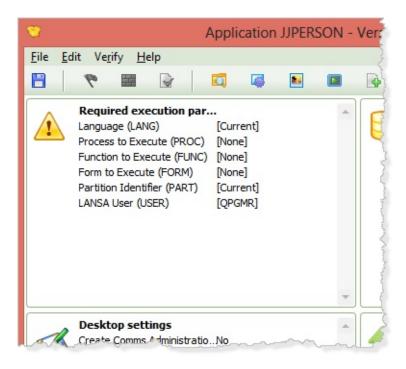
- 10. Select the option: *Include data (new files only)*.

 Including the files in the package means that the OAMs for the files will also be added to the package.
- 11. Click *OK* to close the *Select repository objects* dialog.
- 12. Save the package.

Step 3. Define Execution Parameters

In this step you will define the execution parameters which determine the form, partition, language and user which will be used to start the application for the end user. These are the X_RUN parameters, which is the program which loads the application DLLs.

1. In the *Package Maintenance* dialog, define each Execution parameter by double clicking on the parameter in the *Required Execution* parameters group box.



2. Double click on *Form to Execute*.



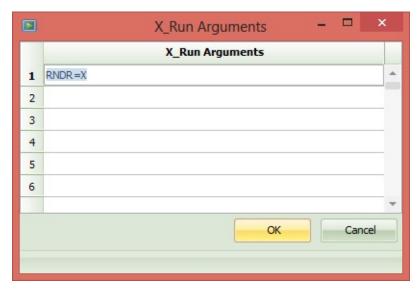
- a. Enter your form name. Note that these parameters may be prompted during installation if required.
- b. Close the *Form to Execute* dialog.
- 3. Follow the same procedure to define the following parameters, making any changes required for your system and national language.

Your trainer will provide alternative information if required. You are deploying to the partition you are using for training.

For training purposes use the current partition as the target for deployment.

User: = **<user id>** where **<**user id> is your LANSA user id.

- 4. The Employees Application must be executed as a DirectX application. This requires a change to the default *Render Type* X_RUN parameter. The *X_Run Arguments* toolbar button opens a dialog which enables additional X_RUN parameters to be defined.
 - a. Open the X_ $Run\ Arguments\ dialog\ and\ add\ an\ entry,\$ **RNDR=X**.

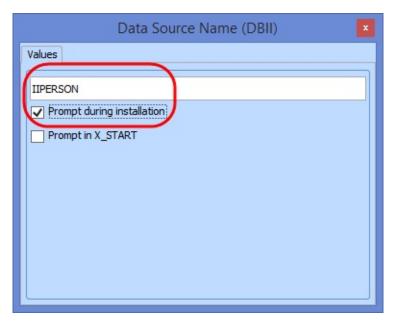


- b. This will execute the application using Direct X.
- 5. Click *OK* to save the changes.

Step 4. Define the Database Parameters

1. In the *Database* group box, open each of the settings detailed below. Set the values shown, adjusting for your application and database.

Data Source Name = **IIPERSON** (i.e. your application name)



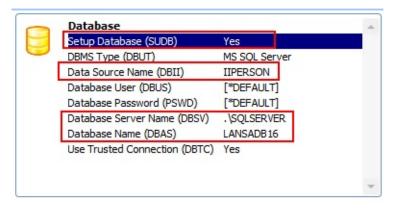
Select the *Prompt during installation* checkbox. This will allow changes to be made during the install if required. You must also select *Display DBMS Dialog* in the *MSI Installation Settings*. The XALONE template sets the *Display DBMS dialog* to 'Y'.

Setup Database (SUDB) = **Yes**

Database Server Name = **.****SQLSERVER**. The database server name was displayed when you connected in *SQL Server Management Studio*, in *Step 1*. e.g. <PC name>\SQLSERVER

Database Name = **LANSADB11**: the name of the new database which you created in *Step 1*.

Leave all other entries with their default values.



Select the *Prompt during installation* checkbox for each of the above parameters.

The database setting are likely to change when installing on different machines.

2. Click *OK* to save the changes.

Step 5. Review the Package Settings

1. Click the *Settings* we toolbar button to open the *Package Settings* dialog.



2. These are the settings defined by the XALONE template which was used to define this package.

Notice that because this package is for a Stand Alone Windows application:

The *Deploy Execution Database Support* option is selected.

The *Deploy LANSA Communications* option is not selected.

To learn more about these settings, refer to Options and Settings in the LANSA Application Deployment Tool Guide.

To learn more about these settings, refer to Options and Settings.

3. No changes are required to these settings.

Step 6. Build Version 1

- 1. *Save* the package definition.
- 2. Check the Package. This is an optional step which will identify any errors in the current package definition, before building the package. Errors will be reported as *Warning* or *Fatal*.



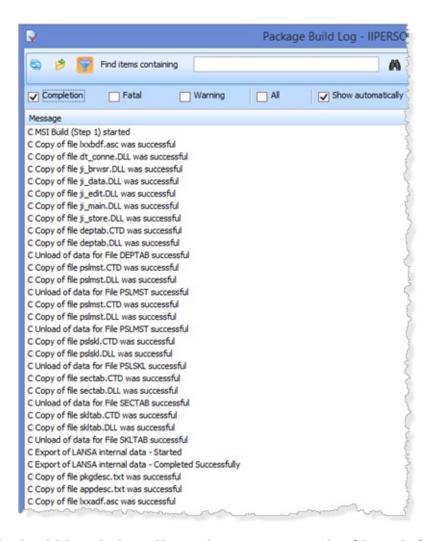
This message box is shown when no errors are found.

3. Build the Package. A progress bar is displayed while the package build steps are completed. This may take some time.



The first time that a new package is built a Global Unique Identifier (GUID) is generated for the package and stored in a file. A message box is displayed which reports details for the GUID which has been generated. For example, folder:

- ...\x_apps\KKPERSON\X_PKGWRK\Guids.txt
- 4. When the build is completed, display the build log and filter for *Completion* messages..



- 5. Notice the build log dialog allows the messages to be filtered, for example by selecting *Completion*, *Fatal* or *All* messages only to be displayed. You search for specific entries in the log using *Find*.
- 6. Scroll down to find the **copy DLLs file** messages. Notice the shipped DLLs includes the four reusable parts which the main form (copy of II_MAIN) depends upon. The DLLS for the employee file (PSLMST) and the files which it also depends on due to file lookup validation rules and predetermined join fields. These DLLs are the OAMs and should include DEPTAB, PSLSKL, SECTAB and SKLTAB. The file data has also been included for these files.
- 7. Check if any *Fatal* messages exist. *Warning* messages may be ignored, depending on the specific message. For example a warning which concerns an image file not found may be ignored, since the image is included in the Image object DLL which is referenced in the form or reusable part.

If fatal messages are reported you will need to review previous steps to resolve the error. For example, if a component DLL is not found it may mean components are not compiled.

```
xtension.dll" -loc "C:\Program Files (x86)\LANSA_T13\X_Win95\X_Lansa se\OdbcDlg.wixobj" "Obj\Release\ServerDlg1.wixobj" "Obj\Release\Ser C MSI Build (Step 2) Complete

39 Completion messages

26 Warning messages

0 Fatal messages
```

- 8. Close the *Package Build* log.
- 9. Close the *Package Maintenance* dialog.

Summary

- How to define a package for a standalone Windows application.
- How cross referencing can be used for deployed components and files.
- The options which the XALONE template includes.

DTE055 – Install Version 1

Objective

To install Version 1 of the Stand Alone Employees Application.

To achieve this objective, you will complete the following:

Step 1. Locate the Install File

Step 2. Install Version 1 of Employee Application

Summary

Deploying Your Own Standalone Applications

It is important to understand that the way the exercises will manage package definition, build and installation may be very different to your own applications. For example, your deployment package should be built on a special deployment copy of Visual LANSA, which ideally should be on another PC or at least in a special install of Visual LANSA on a developer's PC. The objective here is to ensure that all the required components have been identified and are all at the required level.

Installation of Version 1 for your own system could be to an end user desktop or to a file server or to a Terminal Server, depending on how you deploy your client application software.

Refer to the section *About the Tutorials* at the start of these exercises for notes on a real Windows deployment versus this standalone example.

Since you are deploying a standalone application to a new database, you need to deploy both the application, the files it uses and the file data.

In this exercise you will deploy a standalone Windows application to your desktop. It will be installed as a per-user installation. This means it is installed to a path which looks like this, substituting your values:

C:\Users\<user>\AppData\Local\Apps\<company name>\<application name>\

Only the current user will be able to run the application.

If you are deploying your own standalone Windows application, it would usually be installed for all users.

In this exercise, if you are using partition TRN for training, you will deploy the application to a new database with a partition TRN.

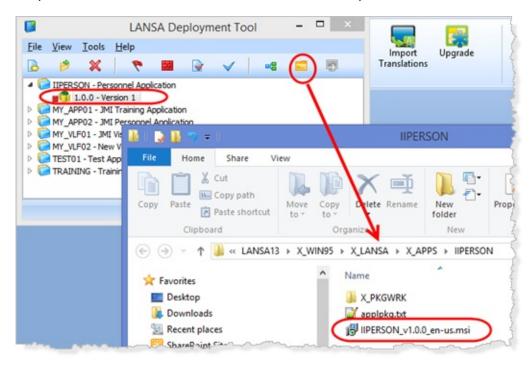
Before You Begin

You must have completed:

DTE050 – Create Version 1

Step 1. Locate the Install File

1. Select your Version 1 standalone Personnel System. Click on the *Go to folder* loolbar button to open Explorer in the deployment tool application folder (in the Visual LANSA ..\X_APPS folder).



- 2. Using your initials the deployment application folder is IIPERSON.
- 3. Using your initials, The Version 1 Windows install file (MSI) will be: IIPERSON_V1.0.0_en-us.msi

Step 2. Install Version 1 of Employee Application

1. The installer must be able to update the SQL Server database.

For Windows 7, your user profile must be in the Administrator group. You can then install by double clicking on the MSI file.

For Windows 8, start a *Command Prompt* using *Run as Administrator*.

Position to the folder: C:\Program Files (x86)\LANSA\x_win95\x_lansa\x_apps\IIPERSON.

2. Enter the following command and then press the *Tab* key. msiexec.exe /I II

where **II** are your initials.

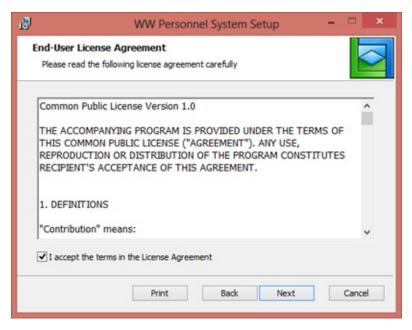
The MSI file name will be completed for you. Your command should now look like this:

Msiexec.exe /I IIPERSON_V1.0.0_en-us.msi

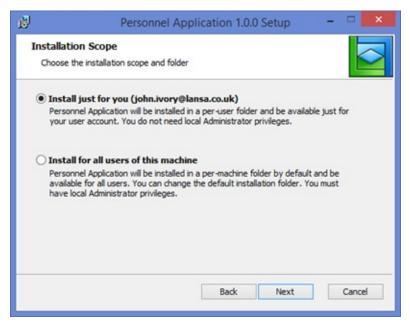
Press **Enter**. The *Windows Setup* dialog will be displayed:



- 3. Click *Next* to continue.
- 4. The End User License Agreement dialog is displayed



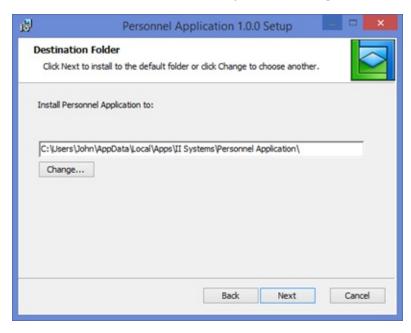
- a. An example license file is supplied in folder:
 - ...\x_apps\x_Wix\Content\license.rtf
- b. You should edit or replace this file to implement your company licensing terms if necessary.
- 5. Select the "I accept" check box and click *Next* to continue:
- 6. You can now select the Installation Scope. Select Install just for you.



Both options will be shown in Windows 7 provided the installer is an Administrator and in Windows 8, provided the installation was started from a

Command Prompt using Run as Administrator.

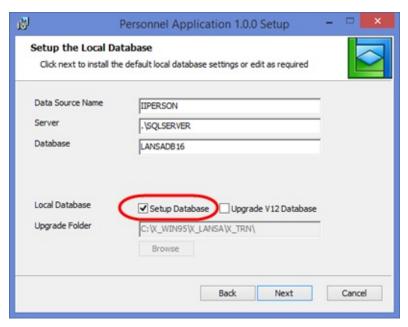
7. Click *Next* to continue. The *Destination folder* dialog will be shown:



The *Change* button allows an alternative folder to be selected.

Notice that the suggested path includes the **Company Name** and **Application Name**:

- ...\II Systems\II Personnel System Standalone\
- 8. Click *Next* to continue. The *Setup the Local Database* dialog is displayed.



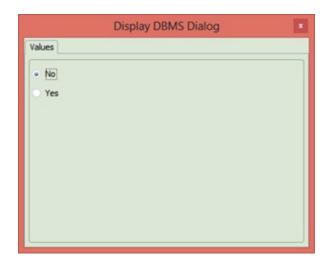
Notice that the parameters which were set to "prompt during installation"

could be changed if required.

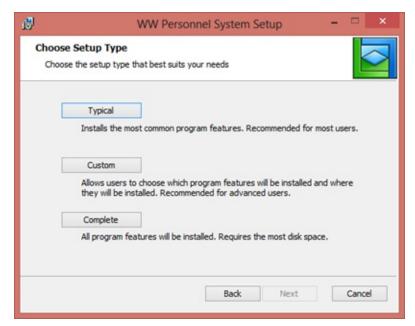
This installation must update the database to define the required application files and insert the file data. This means that the *Setup Database* check box must be selected.

An ODBC DSN entry is created named IIPERSON, pointing to the database server and database name defined by the package.

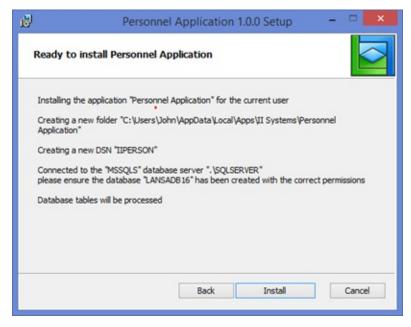
There is an option to suppress this dialog in the package definition. But it is rare that you would not need to offer the choice to Setup the Database. If needed you can suppress this dialog by changing *Display DBMS Dialog* to **No**:



9. Click *Next* to continue. The *Choose Setup Type* dialog is displayed:



10. Select any option to continue. Cuurently, all setup types install the complete package.



11. Click *Install* to continue.

The *Installing* . . . dialog will show progress with messages and a progress bar:

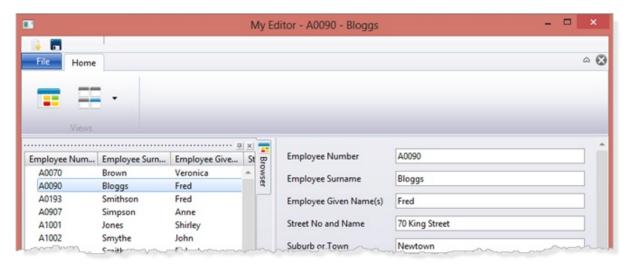


12. Click *Finish* to complete the installation:



Note that the *Start II Personnel System* check box is selected.

The form will run the standard Employees Application as a Stand Alone Windows application.



- 13. As per the application you tested earlier, you can now edit and save an employee record or create a new employee. Double click an employee in the browser list to open it in the editor. A number of employees may be opened, each in their own editor component instance.
- 14. Select *Version 1* for the stand alone *Personnel Application* and click the *Indicate the selected version or patch has been distributed* toolbar button. Version 1 can now no longer be opened or changed.



The *View all objects included in the installation* toolbar button enables *Selected Objects* and *Cross Reference Objects* in the package to be reviewed.

Summary

What You Should Know

• How to install a standalone Windows application as a per-user installation.

DTE060 – **Modify the Employees Application**

Objective

To make a simple change to the Employees Application which will require a Patch to be created to update the deployed application.

Before You Begin

You must have completed the following:

DTE010 – Set Up the Deployment Tool

DTE045 – Create a Stand Alone Application

DTE050 – Create Version 1

DTE055 – Install Version 1

Steps

The changes required are covered by the following exercises.

Complete the following exercises in the *Deploying Client Server Applications* section:

DTE030 – Modify the Employees Application

DTE035 – Create a Patch for the Employees Application. This exercise includes installing the patch.

Summary

What You Should Know

- How to create a patch following to deploy changes to the application.
- A patch is based on the previous version.
- Product Version and/or File Version must be incremented to deploy changed components.

DTE070 - Create Version 2

Objective

To make a minor change to the appearance of the application and then create Version 2 to install the changed application.

Before You Begin

You must have completed the following:

DTE010 – Set Up the Deployment Tool

DTE045 – Create a Stand Alone Application

DTE050 – Create Version 1

DTE055 – Install Version 1

DTE060 – Modify the Employees Application

Steps

The steps for this exercise are in:

DTE040 – Create Version 2 in the *Deploying Client Server Applications* section. Ignore reference to the form II_CONNECT which is not used by this stand alone deployment.

Summary

What You Should Know

- How to create a new version of an application following changes made to the application.
- A new version is based on the previous version.
- Product Version and/or File Version must be incremented before recompiling the changed application. Usually *Product Version* would be incremented.
- A new version un-installs an existing application and replaces it with the new version.

Deploy Applications using JIT

What is Just in Time (JIT) Deployment?

All packages associated with an application are installed on an *Application Server*. That is, the initial version and any subsequent patches or new version. After the initial installation, each time a user launches the application, the application first connects to the *Application Server* to check for any patches or a new version. Any new patch or version will automatically install before the application launches.

The *Application Server* is simply a Visual LANSA system, with a Listener and a SuperServer licence. A single user licence is usually sufficient as it is only used during the JIT Upgrade processing.

The application install package and all subsequent patches or new versions are built as usual but with the addition of a Just in Time setting that points to the JIT server. If the JIT server is on Windows, the only JIT settings required are *Partner LU Name* and *User ID* and *Password*. The User ID and Password must be for a local or domain profile for the server being used.

The exercises required for this deployment are:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE075 – Create a Just in Time Server Installation

DTE080 – Create a Client Install Package with JIT Update

DTE045 – Create a Stand Alone Application

DTE050 - Create Version 1

DTE085 – Install Client Application with JIT Update

DTE090 – Create a Patch for Client Personnel Application with JIT Update

DTE075 - Create a Just in Time Server Installation

Objective

To set up a Just in Time server.

To achieve this objective you will complete:

Step 1. Create a JIT Server Package

Step 2. Install the JIT Server Installation

Summary

Before You Begin

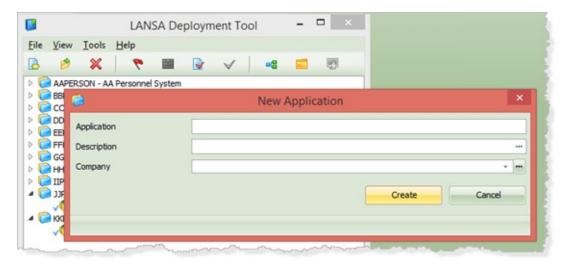
You must have completed the following:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

Step 1. Create a JIT Server Package

- 1. Open the *Deployment Tool* from the *Tools* ribbon.
- 2. Using your initials for **II**, create an application: **IIJITSRV**:

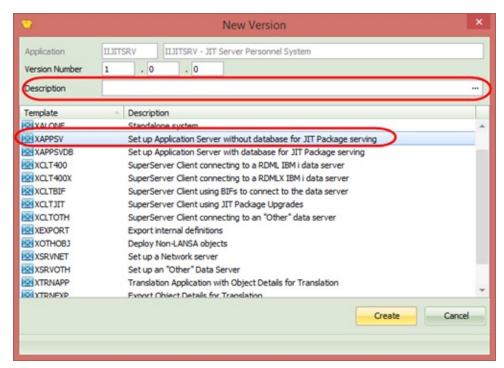


Application: **IIJITSRV**

Description: JIT Server Personnel System

Company: II Systems

3. Click *Create* and the *New Version* dialog will open:



- 4. Enter a *Description*: **Version 1** and select the **XAPPSV** template.
- 5. Click *Create*. The *Package Maintenance* dialog will open.

The XAPPSV template has configured most of the required settings. Settings groups which are not required have been hidden. Note that *Install Listener:* **Yes** is selected.

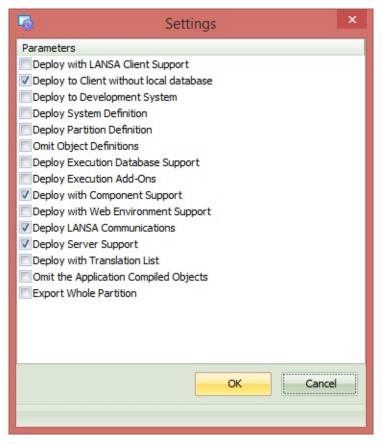


The *Required Execution* parameters include the special value *licences for Process to Execute. This will start the *Server Licence* application to enable a SuperServer licence to be loaded initially.

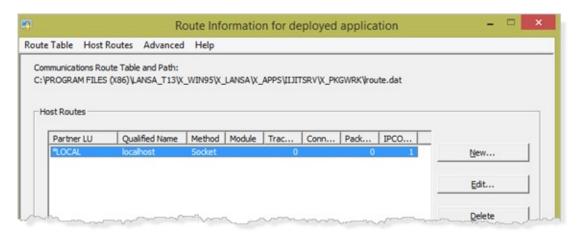
No changes are required to the *Required Execution Parameters* settings.

6. Click on the *Settings* button in the toolbar to open the *Settings* dialog.

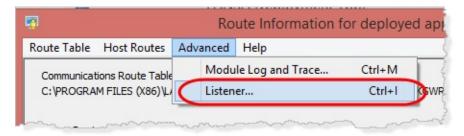
Note: The deployment is to a client without a local database. You would only require a database for a JIT server installation if this system is also the application / database server for the application system. In this case the server would run the applications as well as delivering upgrade packages to the desktop client part of the application.



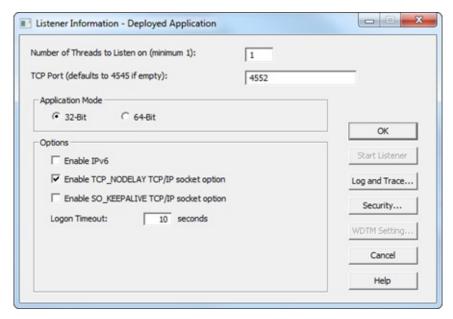
- 7. Click *OK* to close the *Settings* dialog.
- 8. *Save* the package definition using the toolbar button. Close the *Package Maintenance* dialog.
- 9. The JIT Server will install a LANSA Listener. Before building the package, you need to complete the listener definition by selecting the IIJITSRV application and opening the *Communications Administrator* dialog using the toolbar button.



10. Open the Listener dialog from the Advanced menu.



11. Set the listener *Connection Identifier* (it's a Port number). A suitable value will depend on what other listeners you already have defined on the target PC or server. This exercise will install to your desktop. You probably already a your Visual LANSA listener set to the default setting of 4545. For the JIT Server set the value to 4547 or any other suitable value.



- 12. Click *OK* to close the *Listener Information* dialog.
- 13. Close the *Route Information* dialog. This will save a listener dat file in your package definition which will be used to configure the listener when it is installed.
- 14. This change will prompt a message box, as shown:



You have completed the definition of your JIT server package.

- 15. Select the Version 1 package in your IIJITSRV application. Check the package definition with the errors.
- 16. Build your package using the **E** toolbar button.



After the build is completed, the Package build Log will be displayed.

17. Filter to show *All* messages only and scroll to the bottom.



You should have no fatal messages. The one warning message concerns the allocation of a GUID to your package.

18. Close the *Package Build* Log.

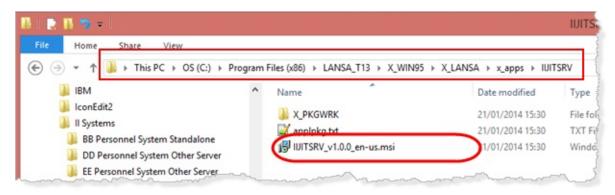
Step 2. Install the JIT Server Installation

For you own JIT Server installation, you would install to a Windows server which is part of your office network and accessible for all-user desktops. For the purposes of this exercise you will install to your desktop.

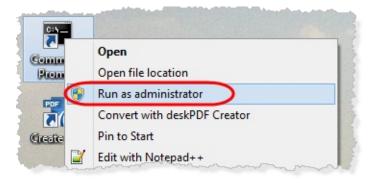
1. In the Deployment Tool main dialog, select the JIT Server application Version 1 and use the *Go to folder* □ toolbar button to open *Explorer* in the ...\x_apps\IIJITSRV folder within your Visual LANSA folder. For example:

C:\Program Files

(x86)\LANSA_T13\X_WIN95\X_LANSA\x_apps\IIJITSRV



- 2a. For a Windows 7 PC, double click on the MSI file to start the installation. Your Windows profile must be a member of the *Administrator* group.
- 2b. For Windows 8 PC, start a command prompt using *Run as Administrator*:



Navigate to the IIJITSRV folder, for example:
 C:\Program Files
 (x86)\LANSA\X_WIN95\X_LANSA\X_APPS\IIJITSRV

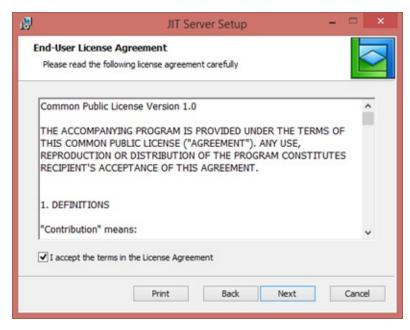
4. Enter msiexec.exe /I II (where II is your initials) and press *Tab*. The MSI

file name will be auto-completed. For example: msiexec.exe /I IIJITSRV_v1.0.0_en-us.msi

- 5. Then press *Enter*.
- 6. The Microsoft Windows installation program will runq, processing the MSI file. A series of messages and dialogs will be displayed.

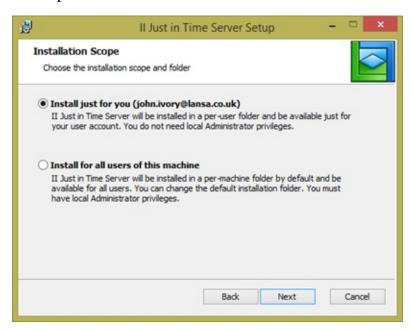


7. Click *Next* to continue:



A sample RTF file is provided, containing a licence agreement. If required you can edit or replace this file to suit your company requirements.

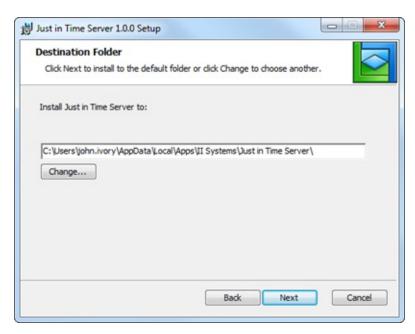
8. Select the "I accept . . ." check box and click *Next* to continue.



9. Select the setting, *Install just for you*. This is suitable for training purposes.

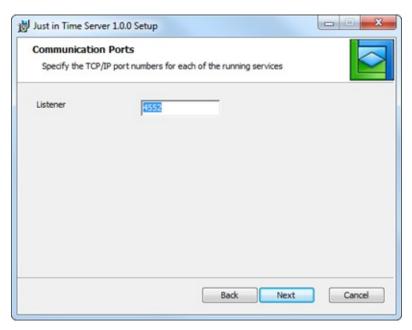
Note: Your own JIT server install would be to a Windows server and would need to be installed for all users.

10. Click *Next* to continue:



If required, the install path could be changed by using the *Change* button to select a different folder.

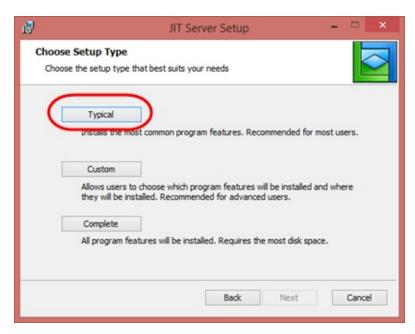
11. Click *Next* to continue:



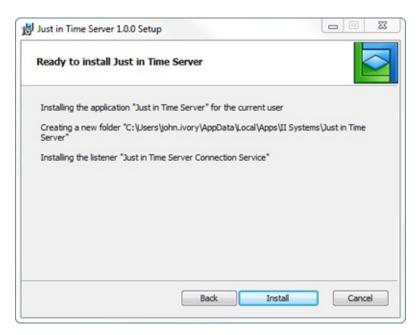
12. The *Communication Ports* dialog will be prompted for the JIT Server Listener. This is always prompted and overrides the value entered in the package definition.

Enter a suitable value for your PC.

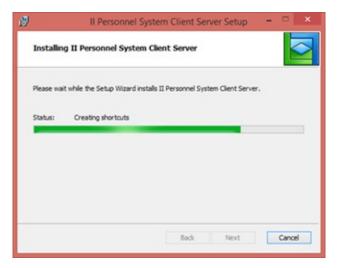
13. Click *Next* to continue:



14. Select *Typical* and then click *Next* to continue.



If required the back button could be used to return to a previous dialog. 15. Click *Install* to start the installation.

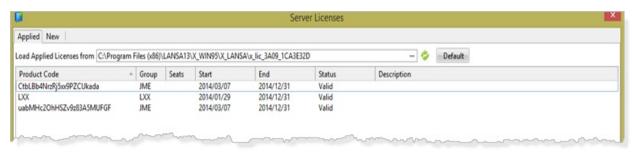


A status bar is displayed while the installation continues.



16. When the install is completed, click the *Finish* button to close the install program.

With the *Start JIT Server* checkbox selected, the *LANSA Server Licensing* application will be started.



The JIT Server requires a single user SuperServer licence to operate. Use the *New* tab to install a licence if you already have one. A License can be obtained via your LANSA vendor. See the LANSA Support web site for details on product licensing. (http://www.lansa.com/support/licensing/)

The X_CPU utility in your Visual LANSA, *Settings and Administration* folder is used to generate a licence request.



The licence is supplied as an XML file, which is loaded via the *New* tab in the *Server Licensing* program. The licensing program records where the licence is installed so that it is accessible for any LANSA install on this machine.

The licence is only valid for the machine matching the X_CPU folder details



- 17. Your Just in Time server installation is now ready to be used.
- 18. In the main *Deployment Tool* window, select Version 1 for the IIJITSRV Just in Time Server application and click the *Distributed* toolbar button. The package objects may now be viewed using the *View all object included in the package* toolbar button, but the *Package Maintenance* dialog may not be opened.

Summary

What You Should Know

- How to create a Just in Time Server package using the XJITSRV template.
- How to define the Listener Port Number.
- The JIT server requires a SuperServer licence. A single user licence will usually be suitable.
- Any other Windows software distribution package can also be used to provide just in time updates.

DTE080 – Create a Client Install Package with JIT Update Objective

To create an installation package for the client server Employees Application which was created in DTE015. The installed software will include settings to access the JIT Application Server. Each time the application is launched the JIT Server is checked for packages. For example, if a patch has not been installed on this PC, then the patch is applied before the application is launched.

To achieve this objective you will complete:

- Step 1. Define Version 1 Package
- Step 2. Define X_RUN Arguments
- Step 3. Define Required Execution Parameters
- Step 4. Define the Just in Time Parameters
- Step 5. Review the Package Settings
- Step 6. Add Routing Table to the Application
- Step 7. Build the Install Package
- Step 8. Copy Client Application Package to JIT Server

Summary

Before You Begin

You must have completed the following exercises:

DTE075 – Create a Just in Time Server Installation

Step 1. Define Version 1 Package

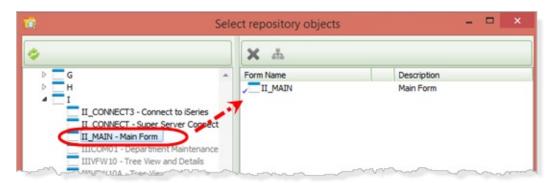
1. Create a new application in the Deployment Tool:

Application IICLTJIT

Description II Client Personnel System with JIT

Company II Systems

- 2. Click *Create* and the *New Version* dialog will be displayed:
- 3. Enter a Description of **Version 1** and select the **XCLTBIF** template. The *Package Maintenance* dialog will open.
- 4. Click the Repository Objects button in the toolbar. In the *Select repository objects* dialog, expand *Forms* and select your main form (copy of II_MAIN). You can add it to the package by drag and drop, or by double clicking it.



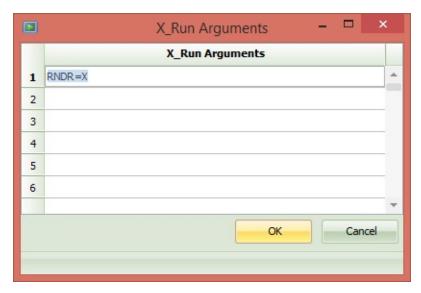
Note the tick shown beside the form. In DTE010 Step 2. Define Default Cross Reference Settings, you defined default cross reference settings for forms and reusable parts, which will automatically include all components used by this main form, and drill down to include their dependents.

5. Click *OK* to close the dialog.

Step 2. Define X_RUN Arguments

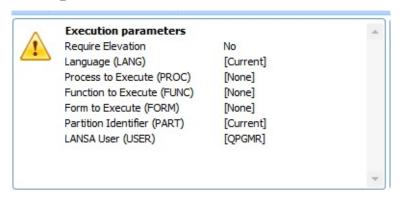
The client application uses DirectX, which means that the X_RUN, which is the executable which loads your applications DLLs, requires a RNDR=X parameter.

1. Click the \square X_Run Arguments button in the toolbar.



2. Click *OK* to save the change and close the dialog.

Step 3. Define Required Execution Parameters



1. In the *Required execution parameters* group, double click each value shown below, to set a value, using your initials:

Form to Execute II_MAIN

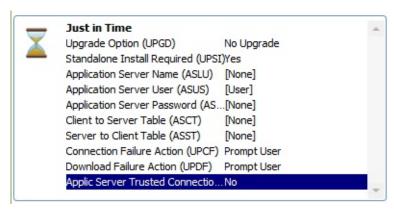
LANSA User Your LANSA profile

Note:

The default settings ensure that the current *Partition* and *Language* will be used.

Require Elevation should be set to **Yes** if this install needs Administrator privileges to run.

Step 4. Define the Just in Time Parameters



1. In the *Just in Time* group, again double click on an entry to set its value as shown below:

Upgrade Option New Versions/Patches

Application Server Name JITSERVER

Application Server User A local user for the PC or server running the JIT

Server

Application Server Password for the above user.

Password

2. Click Save in the toolbar.

Step 5. Review the Package Settings

1. Click the Package Settings button on the toolbar



2. The XCLTBIF template has defined a package which will:

Deploy to client without local database

Include System definition, Partition definition, omits Object Definitions

Deploy with Component Support

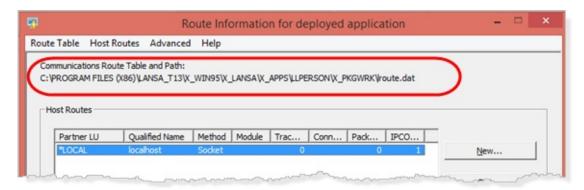
Deploy LANSA Communications

3. Click *OK* to close the dialog. No changes are required

Step 6. Add Routing Table to the Application

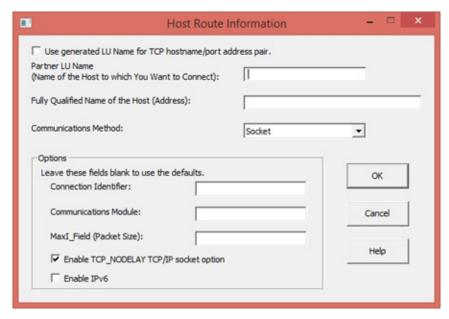
The installed client application will need two routing entries for LANSA communications in order to connect to both the JIT Server and the IBM i Data/Application server.

1. Select the application, and click the <a> Communications Administrator on the toolbar.



The default lroute dat file in the package work folder $(...\X_PKGWRK)$ is opened.

2. Click *New* to create entries for the JIT Server and then click *New* again to create an entry for the IBM i server.



3. In the Host Route Information dialog define the JIT Server as follows:

Partner LU Name JITSERVER

Fully Qualified Name If you have a local DNS, enter the name of your PC.

of the Host (Address) Alternatively enter the IP Address for your PC.

Communications

N. f. . 1

Method

Must be left as Sockets

Connection Identifier This must be the port used for the JIT Server listener.

The recommended value was 4547.

4. Click *OK* to close the dialog.

5. Click *New* to create an entry for your IBM i Data/Application server.

6. Enter the following information for your *IBM i Data/Application server*:

Partner LU Name <a descriptive name for this server connection>

Fully Qualified Name of the <the IBM i server name or IP Address> Host

Connection Identifier <the port number used by the LANSA system

Listener>***

*** You can look for this information in your *Visual LANSA / Settings and Administration folder / LANSA Communications Administrator*. Alternatively look at the Listener job log on the IBM i for the correct LANSA system.

7. Close the *Communications Administrator* dialog.

Step 7. Build the Install Package

1. Check the package with the toolbar button. There should be no errors reported.



2. Build **#** the package with the toolbar button.



A status bar is displayed showing progress.

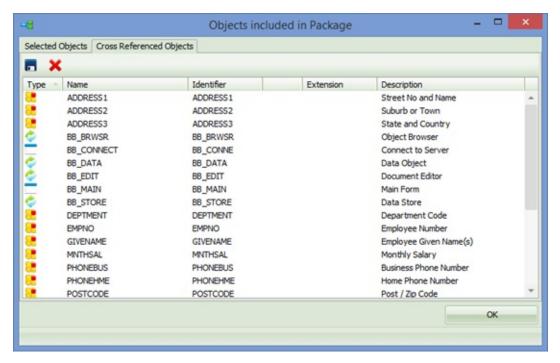
When the package build is complete, the Package Build log is displayed. With *All* messages selected, a summary is shown at the bottom of the log. Alternatively you could filter for *Completion* or *Fatal* messages.



3. There should be no fatal messages shown. If you look for the one warning message you will find this notifies that a unique GUID was generated.

If there more other fatal or warning messages you will need to examine the log more carefully. For example you will have a fatal message if one of your components has not been compiled so that its DLL file could not be copied.

- 4. Close the *Package Build* log. The package is now ready for distribution.
- 5. The *View All Objects* in the Installation ¹⁶ toolbar button displays all the objects included in the package:



Note the *Cross Reference Objects* tab: it shows the objects included by cross referencing.

Step 8. Copy Client Application Package to JIT Server

In the JIT Server installation path you must create an application folder(s) that matches the application(s) to be supported.

The Version 1 installation package must then be copied to the JIT Server **before** the first client installation is carried out. This MSI is not actually installed through JIT, but an MSI that matches the client's MSI is required to be present before a version or patch may be deployed.

- In *Windows Explorer* navigate to the JIT Server installation on your PC. For example, the path could be:
 C:\Users\John\AppData\Local\Apps\II Systems\II Just in Time
 - C:\Users\John\AppData\Local\Apps\II Systems\II Just in Time Server\X_Win95\X_Lansa\X_Apps
- 2. Create a folder with the same name as your application. For example: ...\X_Apps\Iicltjit
- 3. Copy the client application with JIT install package (e.g. iicltjit_V1.0.0_en-us.msi) to the JIT Server **application** path just created. For example:



Summary

What You Should Know

- How to define an application package using the XCLTJIT template, which will check a JIT Server each time the application is launched.
- Understand that an application folder must be created in the JIT Server \x_apps folder containing all MSI and MSP install files for the application to be supported.
- The Version 1 MSI file must be copied to the JIT Server application folder, before the application Version 1 is installed.

DTE085 – Install Client Application with JIT Update

Objectives

To install Version 1 of the client Personnel System application with Just in Time upgrade support.

To achieve this objective you will complete:

Step 1. Locate the Install File

Step 2. Install Version 1 of Client Personnel System with JIT Update Summary

The install steps will be the same as an installation without JIT support.

Once the installation steps are complete, the JIT Server is checked before the application is launched. This could mean, for example, that a user installs Version 1, but as soon as the application is launched, Patch 1 is detected on the JIT Server. This patch would then be installed before the application is launched.

Before You Begin

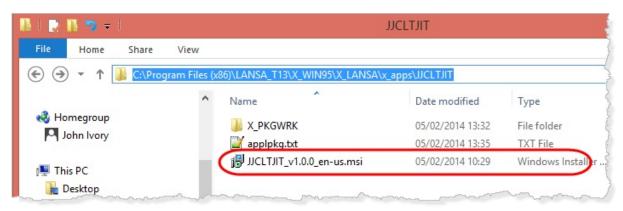
You must have completed these exercises:

DTE075 – Create a Just in Time Server Installation

DTE080 – Create a Client Install Package with JIT Update

Step 1. Locate the Install File

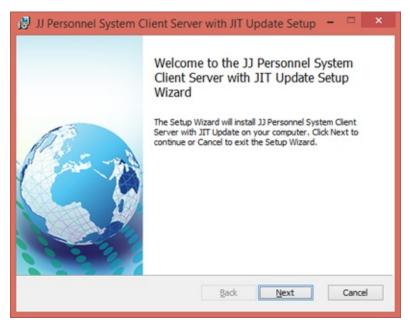
1. In the Deployment Tool, select the *Client Personnel System with JIT Update* application. Select *Version 1* and click the *Go to folder* toolbar button, to open Explorer at this folder:



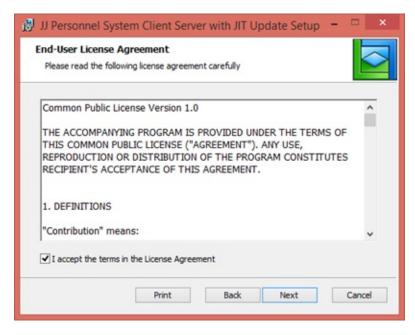
2. Using your initials, The Version 1 install file, will be: IICLTJIT_v1.0.0_en-us.msi

Step 2. Install Version 1 of Client Personnel System with JIT Update

- 1. Double click on the MSI file to start the Windows Install application.
- 2. The Windows Setup dialog will be displayed. Press *Enter*. The Windows Setup dialog is displayed:



- 2. Click *Next* to continue.
- 3. The *End User Licence Agreement* dialog is displayed.

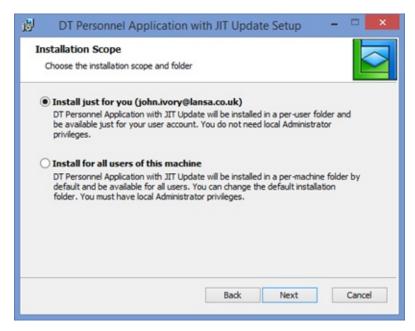


A draft licence agreement is supplied in:

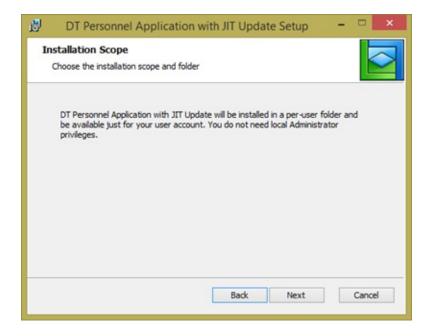
...\x_apps\X-wix\Content\license.rtf

You should edit or replace this file to implement your company licensing terms if necessary.

- 4. Select the "I Accept....." check box and click Next to continue.
- 5. On the *Installation Scope* dialog, select **Install for all users** and click *Next*.



When using Windows 8, the only option as shown below, will be to install as per-user.

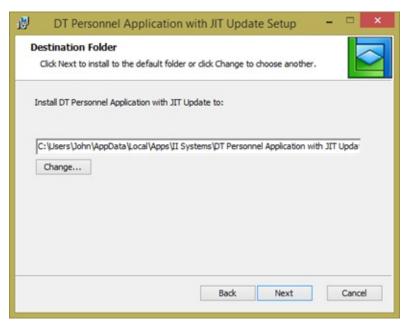


For Windows 8, to install for all users, use msiexec.exe in a command prompt, using *Run as administrator*.

For Windows 7 an all-user installation can be performed using an *Administrator* profile and double clicking on the MSI file.

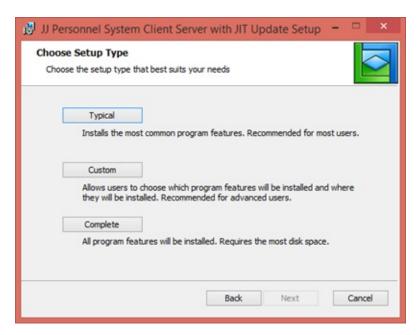
The application will be installed into:

- C:\Users\John\AppData\Local\Apps\II Systems\II Client Personnel Application with JIT Update\X_Win95\X_Lansa\X_Apps\dtcltjit
- 6. The *Destination Folder* dialog allows the destination folder to be changed, if required.

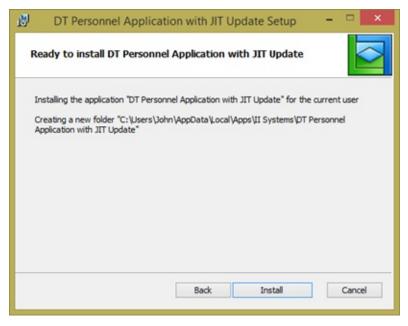


Notice that the suggested path includes Company and Application Name: ...\II Systems\II Client Personnel Application with JIT Update

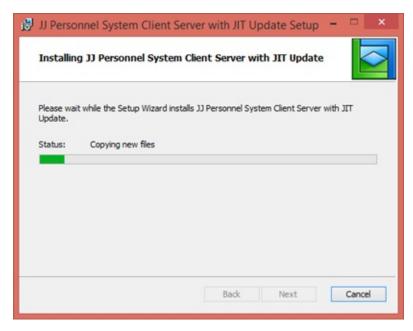
7. Click *Next* to continue. The *Choose Setup Type* dialog is displayed.



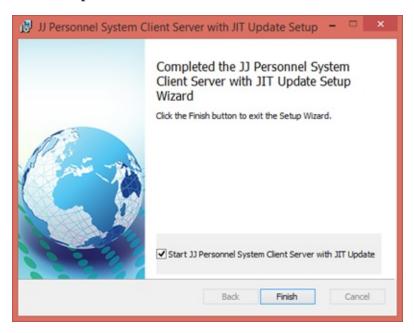
- 8. Click any button to continue.
- 9. Click *Install* to continue.



10. The *Install* dialog will show progress with messages and a progress bar.

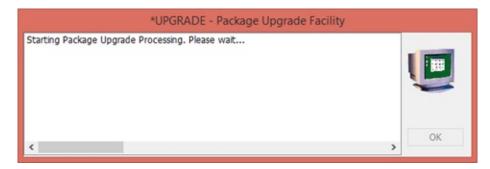


11. Click *Finish* to complete the installation.

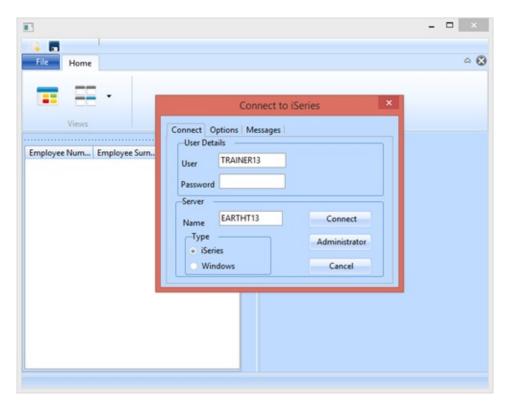


Note that the Start II Personnel System Client with JIT Update check box is selected to run the application immediately.

12. The Package Upgrade process will run before the application is launched. At this stage the JIT Server application folder (\IICLTJIT) contains only the Version 1 MSI file, which means an update is not necessary.



13. The main form (II_MAIN) will run and invoke the SuperServer connector form.



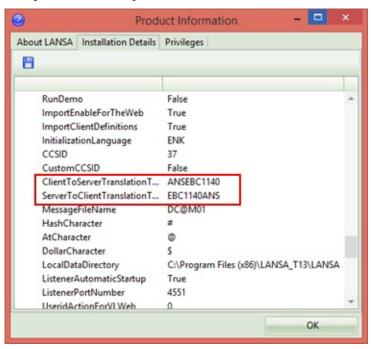
- 14. In the *Connect to IBM i* form, enter your User ID, Password and Server Name.
- 15. Select the *Options* tab, which will initially show the default conversion tables as defined in the connect form. You must enter the correct values for your system, which will depend on your IBM i code page. For example QCHRID = 37 for a US English system.

The EBCDIC to ASCII conversion tables depend on your national language.

You can easily look up this information from your Visual LANSA install configuration:

In Visual LANSA, select *About* from the right hand side toolbar. In the *Product Information* dialog, select the *Installation Details* tab.

Expand the InstalledSettings.cfg entry and scroll down to find the Translation tables for your IBM i system:



Enter these values in the *Options* tab in the *Connect* form.

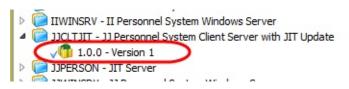
16. Click *Connect*. After connecting with the server, the connect form will close. The main form then populates the browser panel on the left with all employees

Note: The connect form remembers all entries except password.

17. As before, you are now able to edit and save an employee having selected it by double clicking in the browser (left hand side) or using the context menu to Open the employee details in the editor (right hand panel). The application handles multiple employees open in an editor instance.

A new employee may be created using a toolbar button. All changes are saved using a toolbar button. The application warns if you are closing and an employee has been changed but not saved.

18. In the Deployment Tool, select Version 1 for the *Client Personnel Application with JIT Update* application. Click the *Distributed* toolbar button, to change the package status to Distributed. Package Maintenance can now not be opened for this package.



Summary

What You Should Know

• How to install an application package which uses JIT Update

DTE090 – Create a Patch for Client Personnel Application with JIT Update

Objective

To make a simple change to the appearance of the main form and create a Patch 1 and to install via the JIT Server.

Form II_MAIN will be changed by altering its theme (color).

When the application *Client Personnel with JIT Update* is next run, the patch will be automatically applied before the application is launched.

To achieve these objectives you must complete the following:

Step 1. Change Main Form II_MAIN

Step 2. Create a Patch for Client Personnel System with JIT Update

Step 3. Update Application: Client Personnel with JIT Update

Summary

Before You Begin

You must have completed exercises:

DTE075 – Create a Just in Time Server Installation

DTE080 – Create a Client Install Package with JIT Update

DTE085 – Install Client Application with JIT Update

Step 1. Change Main Form II_MAIN

- 1. Open your main form (based on II_MAIN) in the Visual LANSA editor.
- 2. Select the *Source* tab, then select the *Go To* tab, and expand the *Routine* group. Double click the

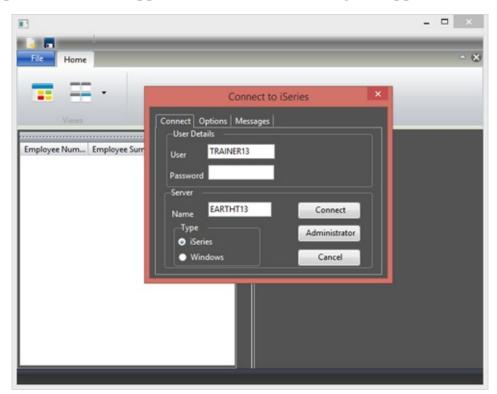
Evtroutine - #com_owner.Initialize

This will position the editor to this routine.

3. Change the Theme to 2007Graphite. Your code should look like the following:

#SYS_sAPPLN.Theme := 2007Graphite

4. Compile and run the application. Check the change of appearance:



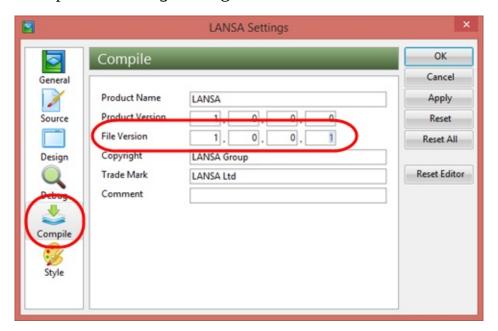
Your simple change represents a developer changing a small part of an application.

In a real application you would now move the changed component(s) to the deployment PC.

Here the changed components need to be recompiled after the *Compile* option settings have been changed to the next *File Version*.

5. This step requires *Partition Security Officer* authority. If necessary, close

Visual LANSA and login again with this profile. From the *File* menu, select *Options* to open the *Settings* dialog.



6. Select the *Compile* settings and increment the **File Version to 1.0.0.1** as shown.

Note: Exactly how **Product Version** and **File Version** are used is your decision. They are a requirement of the Windows Installer. To enable the Deployment Tool to work correctly when building a patch or a new version, you must increment one or both values.

Clearly, one approach would be to increment the Product Version for a new Version and increment the File Version for a Patch.

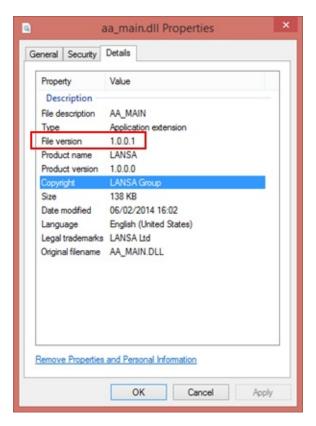
- 7. Click *Apply* then *OK* to close the *LANSA Settings* dialog.
- 8. Close Visual LANSA and login again with your developer profile. Then recompile your main form.
- 9. The new File Version value will be stored in the new DLL file for II_MAIN: Locate the DLL in the following path:

C:\Program Files

 $(x86)\LANSA\X_WIN95\X_LANSA\x_trn\execute$

where **TRN** is your partition.

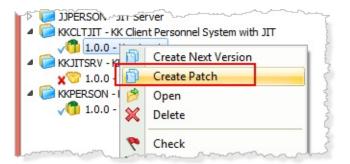
10. Use the context menu (right click) to display the file *Properties*. Select the *Details* tab.



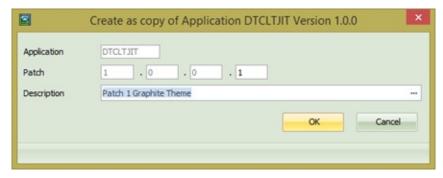
11. You'll see that the File Version is now 1.0.0.1.

Step 2. Create a Patch for Client Personnel System with JIT Update

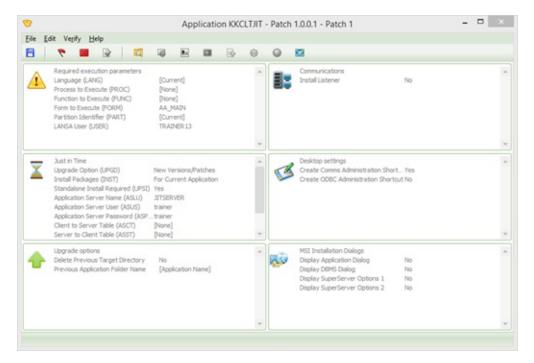
1. Open the Deployment Tool, select the IICLTJIT application. Select Version 1 and use the context menu to select *Create Patch*:



2. Enter a *Description* of **Patch 1 Graphite Theme**. Note that the Patch is numbered 1.0.0.1



The Package Maintenance dialog will open:



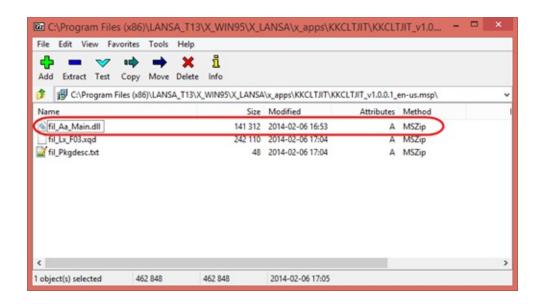
Note that most options and most settings are grayed out. Only the *Selected objects* and *Special file processing* can be modified in a patch. In this case, having changed only the main form (II_MAIN) there are no modifications required to the package.

3. *Save* the package, *Check* it and then *Build* it. When the *Rebuild Cross Reference?* message box is shown, respond **No**, since none of the related components have changed.

Note: If the main form now used a new reusable part, you would need to rebuild the cross reference so the new component is included in the patch file (MSP).

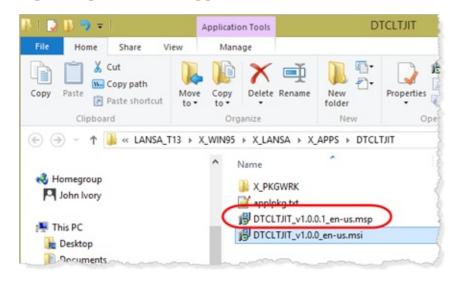
There should be no fatal messages. The Deployment Tool initially builds a new MSI file and compares it with the MSI for Version 1. It then builds an MSP file containing only the changes. In this case, the MSP file will contain only the changed main form II_MAIN.

- 4. Close the Package Maintenance dialog.
- 5. This step is optional and is for information only. If you want to examine the contents of an MSI or MSP file, if you install a suitable zip utility such as 7-Zip, then you can display the contents the file. For example:



Step 3. Update Application: Client Personnel with JIT Update

1. In the Deployment Tool main dialog, select Patch 1 for the *Client Personnel System with JIT Update* application and click on the *Go to folder* toolbar button, to open Explorer in the application folder.



Copy the patch file (e.g. IICLTJIT_V1.0.0.1_en-ms.MSP) to the appropriate JIT Server folder such as:

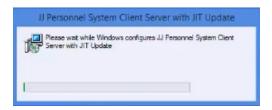
 $C:\Users\In \AppData\Local\Apps\II \ Systems\II \ Just \ in \ Time \\ Server\X_Win95\X_Lansa\X_Apps\dtcltjit$

You are now ready to run the application which will be automatically updated with Patch 1 before the application itself is launched.

- 2. Run the Client Personnel Application with JIT Update.
- 3. The Package Upgrade Facility will contact the JIT Server:



Patch 1 will be applied:



When the update completes, the application will be launched.

4. The application will now use the 2007Graphite theme:



5. In the Deployment Tool select Patch 1 for *Client Personnel Application with JIT Update* application and click the Distributed toolbar button to set the package status to Distributed. *Package Maintenance* can no longer open this package.

Summary

What You Should Know

- How to create a patch for a changed application.
- Understand how to update the JIT Server by copying the install file to the JIT Server's application folder.
- Running the application will check the JIT Server and automatically install the patch.

Deploy to a Windows Server

This scenario demonstrates how to create and install a Windows Data/Application Server system and then execute a client application which connects to the Windows Server application. To do this, you need to create two Deployment Tool applications:

- 1. One application, the one described here, is installed on the server and includes system variables, files (including message files) and their OAMs.
- 2. The second application, described in Deploy Client Server Applications, must be installed on each client and includes the application's forms, processes and functions.

To create the Windows Server installation, you need to perform these steps:

DTE010 – Set Up the Deployment Tool

DTE015 - Create Client Server Application

DTE020 – Create Version 1. In this exercise, in Step 1. Add Routing Table to Application, you need to be aware you are specifying the routing entry to connect to your Windows Server application.

DTE025 - Install Version 1

Exercises DTE015, DTE020 and DTE025 demonstrate how to define and install the client application. This client application will connect to the Windows server installation defined in exercise DTE095 – Create a Windows Server Installation.

DTE095 – Create a Windows Server Installation

Objective

To define a package for a Windows Server installation to provide the Data/Application server used in DTE100 – Execute Client to Windows Server Installation.

To achieve this objective you must complete the following:

- Step 1. Define a new SQL Server Database
- Step 2. Create Windows Server Installation Package
- Step 3. Define the Repository Objects included in the Package.
- Step 4. Define Required Execution Parameters
- Step 5. Define the Database
- Step 6. Define Additional LANSA Features
- Step 7. Check and Build the Package
- Step 8. Install Windows Server Application
- Step 10. Review Windows Server Installation

Summary

Before You Begin

You must complete the following exercises:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1

DTE025 – Install Version 1

Exercise DTE015, DTE020 and DTE025 demonstrate how to define and install the client application. This client application will connect to the Windows server installation defined in this exercise.

Step 1. Define a new SQL Server Database

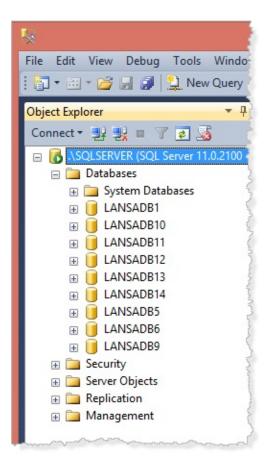
The deployed application will use a new database. The installation package will define all the required tables in this new database. The tables include the LANSA Repository tables as well as the application table and views.

To complete this step you must have *SQL Server Management Studio* installed.

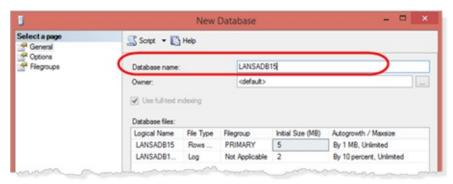
1. Open *SQL Server Management Studio* and connect to the database server service:



2. Expand the Databases group so that you are aware what database names have already been used:



- 3. Right click on *Databases* and select *New Database* from the context menu.
- 4. Enter a suitable name for the new database, for example: LANSADB15.



Click OK to create the new database using the default configuration.

5. Minimize *SQL Server Management Studio*, which you will use again in a later step.

Step 2. Create Windows Server Installation Package

1. Open the Deployment Tool from the *Tools* ribbon and create a new application

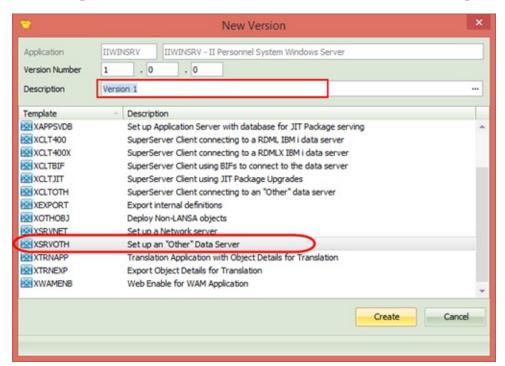
Application IIWINSRV

Description II Personnel System Windows Server

Company II Systems

The *New Version* dialog is displayed.

2. Enter a *Description* of **Version 1** and select the **XSRVOTH** template:



3. Click *Create* to continue.

The *Package Maintenance* dialog will be displayed.

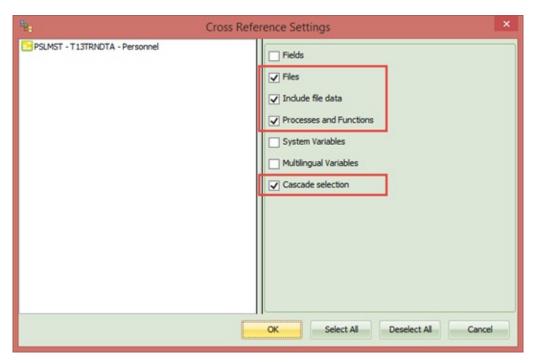
At this point you need to consider how to define the server portion of the Personnel Application. The elements of the application which you need to deploy to the Windows server application will include the following:

- The application files and their data.
- The OAMs for the files. These DLLs will be automatically included when you add the files to the package.

- Any processes and functions which the file OAMs depend on. This will include trigger and validation functions which the OAMs may call.
- Any called server functions which the client application may call. There are none in this simple application.

Step 3. Define the Repository Objects included in the Package.

- 1. Open the *Select repository objects* dialog using the loolbar button. Expand the *Files* group and add the following file to the package: PSLMST Personnel
- 2. Select the file in the right hand list, and click the *Cross Reference* toolbar button.



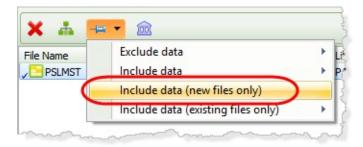
- 3. Select these check boxes:
 - Files
 - Include File Data
 - Processes and Functions
 - Cascade Selection

These options will:

- Include other files used by the selected file.
- Include file data for the files added by cross reference.
- Include processes and functions which are used by the file definitions
- Cascade the selection to other files used by the files added through cross

reference.

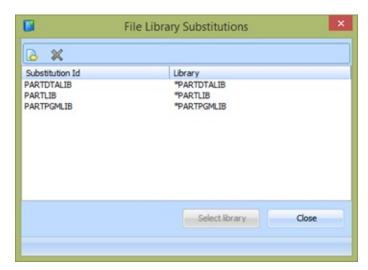
- 4. Click *OK* to save these changes.
- 5. Select the file PSLMST in the list, click on the *File Data* toolbar button:



6. Select the option *Include data (new files only)*. This option will include data for the file, if the file does not already exist in the database.

Note: Typically you would only include data for lookup tables, for examples a table containing a list of USA States. Application data files (e.g. invoices, orders, customers etc) would be empty files at this stage so they are deployed without data.

7. With the file in the list selected, click the *File Library Substitutions* at toolbar button.



8. Select the *PARTLIB Substitution Id*, and click *Select Library*, which will also close the dialog.

Tables in an SQL database are defined belonging to a schema (this term varies depending on the DBMS). LANSA uses the library name to set the table schema name. The selected option will give all files a schema based on

the default file library name for the partition into which they are deployed. For example a library name of T13TRNDTA will generate a SQL Server schema name of X3TRNDTA. The schema name is limited to eight characters.

Note: See 5.2.1. *Mapping a File's Library to a Fixed Deployment Schema* in the *Deployment Tool* guide. This provides a full description of the features available for managing the schema name used in an SQL database.

Note: See *Mapping a File's Library to a Fixed Deployment Schema*. This provides a full description of the features available for managing the schema name used in an SQL database.

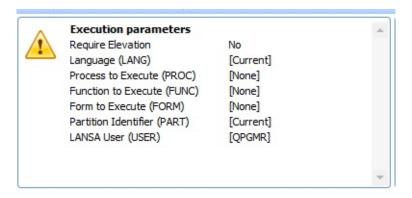
9. Click *OK* to close the *Select repository objects* dialog.

You have added the main application file (PSLMST) and all the other files which it depends upon through validation rules and predetermined join fields.

Step 4. Define Required Execution Parameters

This step will define the parameters needed for the X_RUN program which executes the server application when called by the client application.

1. Set each parameter by double clicking it to open a dialog where you set its value. In many cases you can also define whether this parameter may be set at run time.



Parameters with a value of **[Current]** will take their value from the LANSA partition in which the Deployment Tool is used to define the package. In many cases these parameters can be left as **[Current]**.

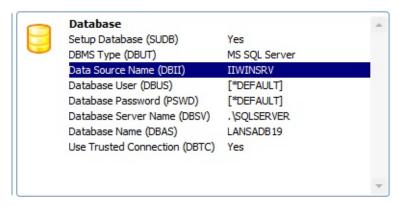
Since this is a server deployment there is no process, function or form to be run initially.

2. Double click *User* and enter your **Visual LANSA user id**. The LANSA job will be identified by this user.

Step 5. Define the Database

This step defines the database attributes which the server application will connect to and use.

These parameters provide further X_RUN parameters for use when any server application program executes.



- 1. Notice that DBMS Type (DBUT) has defaulted correctly to MS SQL Server.
- 2. Enter the following values:

Setup Database (SUDB) Yes

Data Source Name IIWINSRV

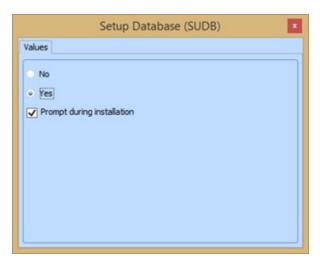
Database Server Name .\SQLSERVER ###

Database Name LANSADB15 ***

This is the default name given to the SQL Server database server service.

*** depends on the name used to define the new SQL Server database.

Select the *Prompt during installation* checkbox for each of these parameters so that they can be changed during installation if necessary. The *MSI Installation Dialogs* group must contain *Display DBMS Dialog* = *Yes* which is true for the template used to create this package.



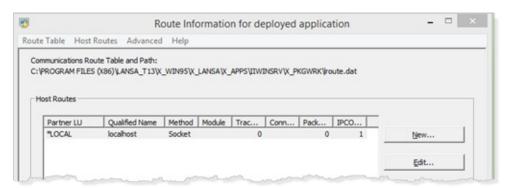
3. If the database is using *Trusted Connections*, then *Database User* and *Password* are not needed. The database connection will require a valid local Windows user id and password.

Step 6. Define Additional LANSA Features

This server installation will need to run a a listener so that the client can connect in order to access the database or run called server functions. The XOTHSRV template used to create this package has *Installer Listener* = *Yes*.



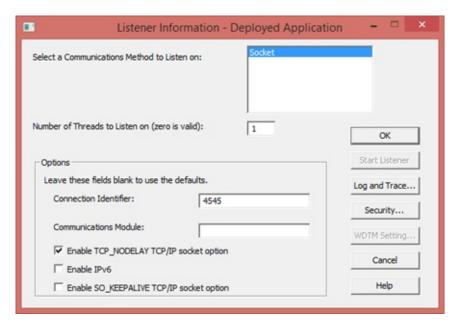
- 1. Save the package definition and close the *Package Maintenance* dialog.
- 2. In the main Deployment Tool window, select the IIWINSRV application. Select the *Communications Administrator* toolbar button.



- 3. The *Communications Administrator* is used in two ways by the Deployment Tool:
 - Defines a routing table with the entries needed by a client to connect to one or more servers. The routing table is file: lroute.dat
 - Defines a Listener, which the package installation will set up on the server. The listener definition is held in a file: listener.dat.

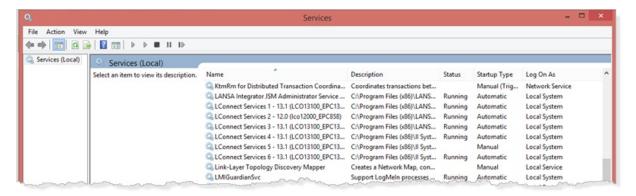
For this server installation you need to define a listener.

4. Select *Listener* from the *Advanced* menu.



The server application listener needs a *Connection Identifier* (port number) which will be unique on the machine it is to be installed onto. In this exercise you will be installing the server application to your own PC. If you have one copy of Visual LANSA installed, you already have one listener, normally using a default *Connection Identifier* of **4545**. If you have completed the *Deploying Applications using JIT* exercises you will have another listener for the JIT server. The suggested port number for this was **4547**.

The LANSA Listener is a local Windows service and if necessary, you can check how many listeners you already have using Windows *Control Panel / Administrative Tools / Services*:



Scroll down to find the *LConnect* entries. The example shown above is unusual. This PC has a number of Visual LANSA installations as well as a number of deployed applications.

If necessary you can find the *Communications Administrator* for each LANSA installation and then find its *Connection Identifier*.

- a. Enter any suitable value in the *Connection Identifier*. In most cases this could be **4546**.
- b. Click *OK* to close the *Listener Information* dialog.
- c. Close the Route Information dialog.

You will see this message box, after defining the Listener information:



Step 7. Check and Build the Package

- 1. Select the Version 1 package. Click the *Check* toolbar button. No errors should be reported.
- 2. Select the *Build* toolbar button. A progress bar will be displayed as the package is being built.

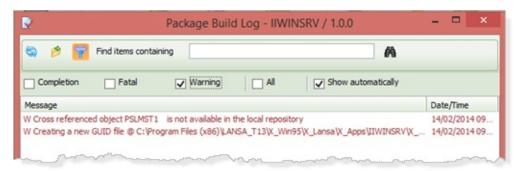


3. When the build is complete the *Package Build Log* will be displayed. Filter for *Fatal* messages. You should have no fatal errors. You can also use the toolbar checkboxes to display only *Completion*, *Warning* or *All* messages.

A **Fatal message** will always mean you need to correct a problem and rebuild the package.

Warning messages should be reviewed, but may not always require further action.

The following example illustrates a warning that requires action:



The file PSLMST – Personnel table appears to have a missing View, PSLMST1. The file probably needs to be rebuilt and the package then rebuilt.

Step 8. Install Windows Server Application

A Windows Server install will always need to be run using an Administrator profile. For your own application you would now install the server application to a Windows server. For training purposes you will install it to your PC.

1. In the Deployment Tool main window, select the IIWINSRV application and click the *Go to folder* toolbar button to open Explorer in the application folder. For example:

C:\Program Files

$(x86)\LANSA\X_WIN95\X_LANSA\x_apps\IIWINSRV$

- 2a. For Windows 7, using a user who is a member of the Administrator group, double click on the MSI file to run it.
- 2b. For Windows 8, open a Command Prompt using Run as Administrator:

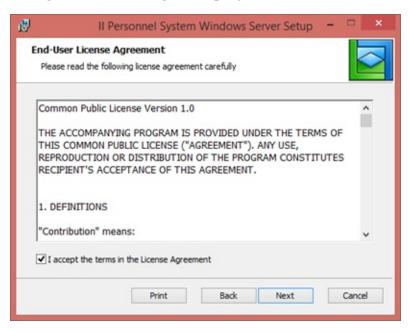
Navigate to the application packages folder shown above.

Enter the command msiexec.exe /I II then press *Tab* and the MSI file name will autocomplete, for example: msiexec.exe /I IIWINSRV_v1.0.0_en-us.msi then press *Enter* to start the install.

3. A series of Windows install dialogs will be displayed:



- 4. Click *Next* to continue.
- 5. The End User Agreement dialog is displayed:

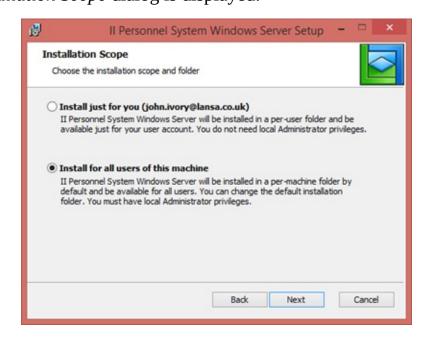


A sample end user licence agreement is provided in:

 $...\x_app\X_Wix\Content\license.rtf$

You should edit or replace this file to implement your company's licensing terms if necessary.

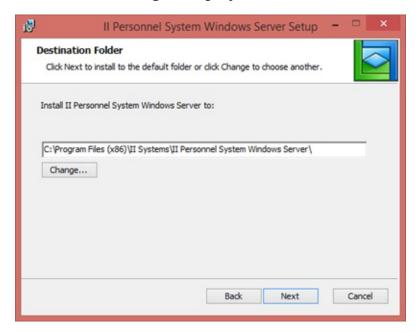
6. Select the 'I accept....' check box and click *Next* to continue. The *Installation Scope* dialog is displayed:



When installing a server application, this should always be installed for *All Users*.

7. Select the *Install for all users of this machine* radio button. Click *Next* to continue.

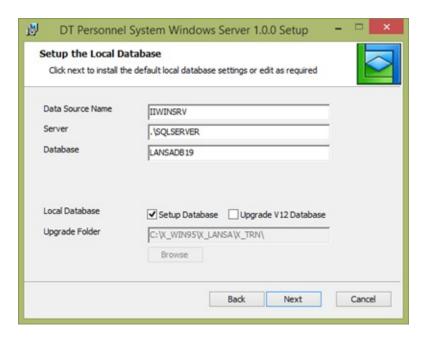
The *Destination Folder* dialog is displayed:



You will notice that the default install path includes *Company Name* and *Application Name*.

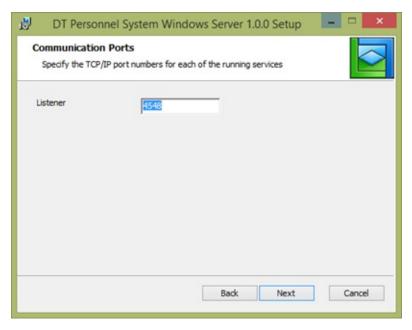
The *Change* button allows you to select an alternative install path.

8. The *Setup Local Database* dialog is displayed:



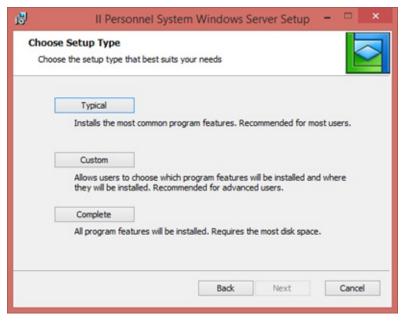
Notice that a number of parameters are input capable and may be changed, because the *Prompt during installation* checkbox was selected.

- 9. It is essential to be aware at this point that the *Local Database*, *Setup Database* check box must be selected. The install will then continue by defining and populating the LANSA repository tables and depending on what options were setup for the application files, the install will build the application tables and views and load the file data.
- 10. Click *Next* to continue.
- 11. Since a Listener is being installed the Communications ports dialog is shown. Confirm that the port number is as required.



12. Click *Next* to continue.

The *Choose Setup Type* dialog is displayed:



13. Select any *Setup Type* and then click *Next* to continue. The *Ready to Install* dialog is displayed:



This dialog shows:

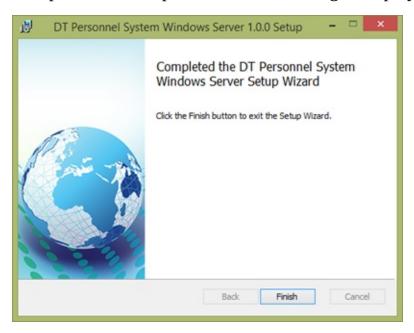
- where the software will be installed
- the ODBC System Data Source Name (DSN) entry which will be created
- the database server connection and database name used
- a LANSA Listener will be installed.

Note that during the install process, each dialog has a *Back* button which allows you to back out and change earlier options which were selected.

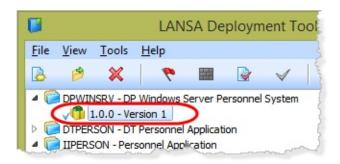
14. Click *Install* to continue and a progress bar will be displayed as the install file is unpacked and run.



Once the install process is complete, the *Finish* dialog is displayed:

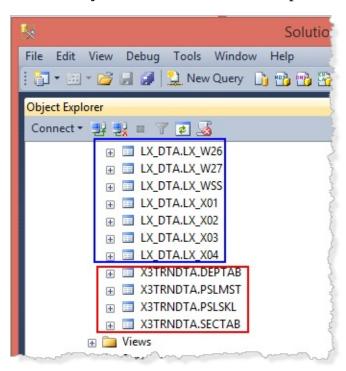


- 15. Click *Finish* to close the install program.
- 16. In the Deployment Tool main dialog, select Version 1 in the II Windows Server Personnel System application and click on the Distributed toolbar button to set the package status to Distributed. The package can no longer be opened in the Package Maintenance dialog.



Step 10. Review Windows Server Installation

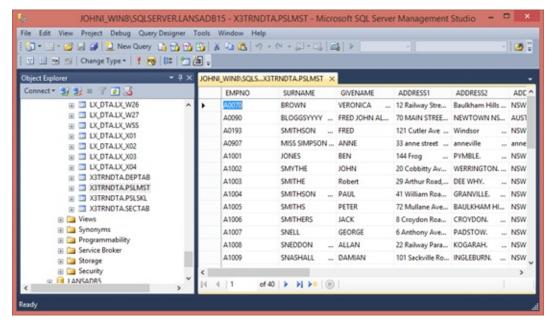
- 1. Switch to *Microsoft SQL Server Management Studio* that was used in an earlier step.
- 2. On the *View* menu, select the *Refresh* option. This should refresh the *Object Explorer* view on the left hand side. If this function seems unreliable, close and re-open the Management Studio application.
- 3. Expand the database that you created earlier and expand the *Tables* group.



You should find a large number of tables prefixed LX_DTA which is the schema name for LANSA repository tables.

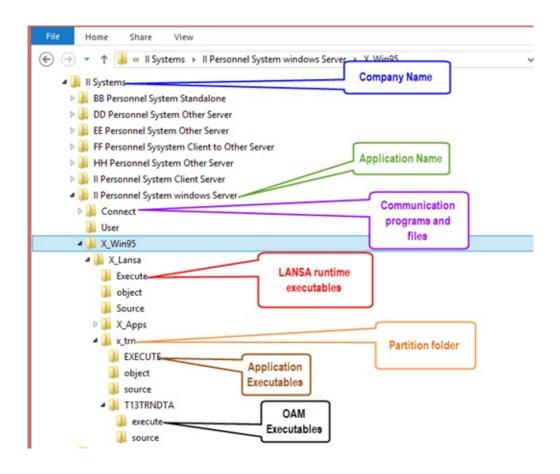
The second group of tables prefixed by X3TRNDTA (schema name) in the sample shown are the application tables. Your schema name will depend on your LANSA partition file library name. Schema name is limited to eight characters so that LANSA shortens the library name if necessary. The library name was T13TRNDTA in this example.

4. Right click on the table PSLMST and select *Edit Top 200 rows*.



The file data should be displayed on the right hand side.

- 5. Open *Windows Explorer* and navigate to the path, where <company name> and <application name> are your values for the Windows Server install:
 - C:\Program Files (x86)\<company name>\<application name>
- 6. The following image shows the main contents and the organisation of the installed folders:



Summary

What You Should Know

- How to define a simple Windows server installation.
- How to install a Windows server application.
- The main structure and content of the installed application's folders.
- Both Windows 7 and Windows 8 will allow *Install just for you...* if the install was run by simply double clicking the MSI file.
- In Windows 8, you must run the install from a command prompt started with *Run as Administrator*, in order to *Install for all users*.

DTE100 – Execute Client to Windows Server Installation

Objective

To execute the client application connecting to the Windows Server Application defined and installed in DTE095 – Create a Windows Server Installation. This exercise simply explains things you should be aware of and the differences with execution against an IBM i server.

Before You Begin

This small exercise explains how to execute your client application against a Windows server installation

To create the Windows server installation, complete:

DTE095 – Create a Windows Server Installation

In DTE095, you will have created the client application explained in:

DTE010 – Set Up the Deployment Tool

DTE015 – Create Client Server Application

DTE020 – Create Version 1. In this exercise, in Step 1. Add Routing Table to Application, you need to be aware you are specifying the routing entry to connect to your Windows Server application, which is actually installed on your PC. our routing entry will need to be similar to:

- Partner LU: IIWINSRV
- Qualified Name: localhost
- Connection Identifier: 4546

The *Connection Identifier* must be the value used to define the listener for the server installation in DTE095 – Create a Windows Server Installation.

DTE025 – Install Version 1. Stop after the last installation step at *Step 2.9* and return to this exercise.

Client Connecting to a Windows Server

You will find more details on this topic in this guide. For example refer to Deploy Client / Server Applications.

If you plan to implement your own application with a Windows or Linux server you must review this material in detail.

Following is a high level summary that covers only the main concepts:

Windows Client Connecting to Windows Server using SuperServer Built In Functions

This is the scenario covered by this group of exercises. It is also the recommended architecture for all LANSA client server applications, whether the server is IBM i, Windows or Linux.

The Windows Server LANSA installation includes:

The database, with Microsoft SQL Server, Oracle, MySQL or Sybase SQL Anywhere as the DBMS. Note that the licensing of this deployed DBMS is your responsibility. See the *Support Platforms* document on the LANSA web site for details of the versions supported.

- The database must contain the LANSA Repository tables and all of the application tables and data.
- Visual LANSA runtime software which supports all the LANSA executables.
- Object Access Modules (OAM) for all the application tables. These are DLLs created by Visual LANSA when each file is compiled.
- The DLLs for all functions that may be called by the OAMs as triggers or validation programs.
- The DLLs for all functions that may be called by the client application (CALL_SERVER_FUNCTION).
- A LANSA Listener communicating with client application.
- A SuperServer (LXX) licence must be installed on the Windows server. See Product Licensing on the LANSA Support web page.

The Windows client installation includes:

- Visual LANSA runtime supporting the application executables
- The client application as DLLs
- LANSA communications software and configuration files (routing table, lroute.dat).

The client may be installed on the desktop, on a file server or on a virtual machine (Citrix or MS Terminal Server).

Starting LANSA Windows Applications

All LANSA Windows applications are launched by one executable - $x_run.exe$. This requires a number of parameters to establish the runtime environment such as partition and language. Refer to the *Technical Reference* for full details of x_run parameters.

The x-run program may be started using $x_{start.exe}$. This executable has one parameter, the name of a text file, which contains the x_{run} parameters. Deployed applications are provided with a shortcut that uses $x_{start.exe}$.

Other ways to provide parameters to x_run include:

- Using Environment variables
- Using the profile file x_lansa.pro

Once again you should see the *Technical Reference* for full details about these topics.

Controlling the Windows server runtime environment

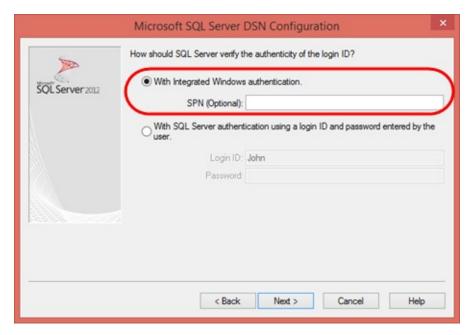
When a Windows client connects to a LANSA Windows server application, the server application inherits its x_run parameters from the client. This is convenient for some parameters such as PART=PRD but other client x_run parameters will be wrong, such as DBII=*none. The DEFINE_ANY_SERVER built in function has an "x_run exceptional arguments" parameter which enables overrides to the parameters the server will use. A special value "*server" can be used here, to ensure the server application starts with the desired parameters, for example, DBII=*server. The SuperServer connect form provided with this set of exercises uses the following x_run exceptional arguments on the DEFINE ANY SERVER.

DBII=*server DBUT=*server DBID=*server

These arguments enable the deployment package for the Windows server application to have Database settings which will then be applied at run time.

Other Considerations

When the client connects to the Windows server application, a database connection must be established on the server. If a Microsoft SQL Server database is configured to use Integrated Windows authentication then the client is simply required to provide a valid Windows user id and password to start the server application and connect to the database.



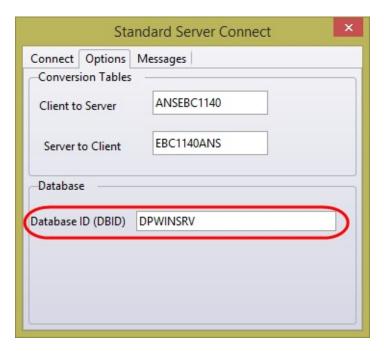
If another DBMS such as Oracle is being used, you will need to be aware of how the database authentication is configured.

Step 1. Run the Client to Windows Server Application

1. Double click on the II Personnel System icon on your desktop. The application's main form immediately opens the SuperServer connect form:



- 2. Enter your **Windows user id** and **password**.
- 3. Enter the **Partner LU Name** for your Windows Server installation. You can use the *Administrator* button to run the *Communications Administrator* to check for the name you gave this connection.
- 4. Select the *Windows* option for *Type* of server.
- 5. Select the *Options* tab and enter the *Data Source Name* you defined in the *Database* group when defining the package. This defines a System DSN entry in the ODBC configuration file. To check this, see your Visual LANSA *Settings & Administration* folder / *32-Bit ODBC Administrator*.



6. Click the *Connect* button. The main form should now load the browser panel on the left hand side, with all employees from the file PSLMST.

The client server connection tables are not used by the Windows DEFINE_SERVER built in function and will be ignored.

Debugging Your Windows Client Server application

If you initially experience problems with connecting to your Windows server application you should do the following:

1 Look for a X_ERR.LOG. This will be written to the path defined by the X_RUN parameter TPTH. The default value for this path is the user's temporary folder which can be found using the shortcut %TEMP% for the path. You should refer to the *Technical Guide* for more information on how this path is evaluated. One or more of the following log files may be found: MSIxxx.log

DPinstal.log

LANSAInstallLog.txt

LANSAInstallPermissions.txt

X_err.log

Import.log

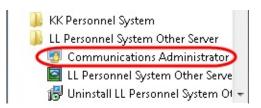
InstallTrace.txt (requires x_run parameter ITRO=Y. This may be set globally in a Windows environment variable)

- 2. Turn on tracing in the LANSA Listener for the Windows server application. See the *LANSA Communications Setup Guide* for details on how to setup tracing. A brief description follows.
- 3. Turn on tracing in the LANSA application. Application level tracing is enabled by the x_run parameter ITRO=Y.

Turn on Tracing for the LANSA Listener.

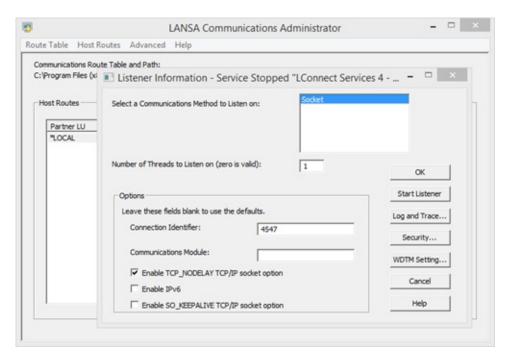
The *Communications Administration* program (lcoadm32.exe) is installed to the Windows server application's \connect folder.

1a. For Windows 7, look in the *Windows Start* menu under your company name and application name. Double click on the *Communications Administrator* program.

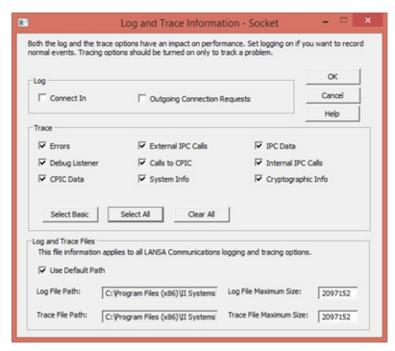


1b. For Windows 8, use *Search* to find *Communications Administrator*, use the context menu to *Run as Administrator*.

a. Select Listener from the Advanced menu

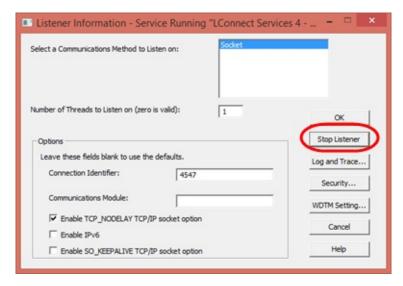


b. Click the *Log and Trace* button.



- c. Use the *Select All* button to show all trace options selected, as shown in the screen capture above. You should refer to the *LANSA Communications Setup Guide* for detailed recommendations.
- d. Click *OK* to close the *Log and Trace Information* dialog.

e. Use the *Stop Listener* button to stop and the start the listener.



2. Run your client application to connect to the Windows Server application. Then examine the trace file which is produced. The listener trace file (lroute.trc) is written to the \Connect folder in the Windows server application folder. For example:

C:\Program Files (x86)\<company name>\<application name>\Connect

Turn on Application Level Tracing

The x_run parameter can most easily be set by defining an environment variable or by adding the parameter to the LANSA profile file (x_lansa.pro).

1. Locate the LANSA profile file (x_lansa.pro) - for example, in the x_lansa folder:

C:\Program Files (x86)\<company name>\<application name>\x_win95\x_lansa

2. Open the x_lansa.pro file with Notepad.

```
File Edit Format View Help

UPCD=6B145DFC-767B-4C15-AEA9-3D88FC455445
PART=TRN
LANG=ENG
DBUT=MSSQLS
DBII="HHPERSON"
DBUS=John
USER=John
PSUS=John
APPL=HHPERSON
SUDB=N
INST=MSI
TPTH=c:\temp
```

- 3. Add an **ITRO=Y** parameter.
- 4. Optionally add a **TPTH=c:\temp** parameter, using any suitable folder name. The default path is the user's Windows temporary folder path (%TEMP%).
- 5a. Save the x_lansa.pro file, ensuring you use *Save As* and select *Save as Type* (*.*) to prevent Notepad adding txt as the file type.
- 5b. For Windows 8, you must start Notepad using *Run as Administrator*. The trace file will be named **x_tracenn.txt** where nnn is a sequence number.

Summary

What You Should Know

- How to run a client application to a Windows server application.
- How to turn on application level tracing.
- How to turn on tracing in the Listener.
- How to find the Visual LANSA x_err.log file.

Cloud Tutorials

The exercises in this tutorial assume that you are an experienced Visual LANSA developer and already have an AWS account and have used it to create Windows instances and remotely access them.

About the Exercises

Introduction

These exercise describe various ways LANSA may be deployed into the Cloud and how you may interact with your LANSA Stack.

The Application to be Deployed

It's presumed that you already have your application packaged into an MSI.

Tutorial versus Real World

The exercises in this tutorial are designed to simplify deployment for training purposes and to minimize the setup involved.

System Requirements for these Exercises

It is presumed that you already have an AWS account. If not, navigate to this url and follow the instructions: aws.amazon.com.

These exercises instantiate Cloud services for which you may be charged. You are responsible for all such charges that you may incur.

Deploy A WAM Application into AWS

This section contains exercises that demonstrate how to deploy a WAM application into AWS. If all you need to do is provision a demonstration of a LANSA Stack, then essentially it's just a few clicks and then wait in the region of 45 minutes for all the parts of the stack to be provisioned.

If you need to deploy your own application, then the application must be uploaded first and more care will need to be taken over the database server type, the size of the virtual machine instances, the number of instances and security settings.

The exercises to be performed to complete this objective are:

CTE015 – Subscribe to the LANSA Image

CTE030 – Upload your LANSA WAM Application

CTE040 – Create the LANSA Stack

CTE050 – Watch the LANSA Stack Scale In

CTE060 – Apply Windows Updates to all Web Server Instances

CTE070 – Apply LANSA Patch to all Web Server Instances

CTE080 – Apply LANSA Application Upgrade to all Web Server Instances

CTE015 – Subscribe to the LANSA Image

Objectives

In this exercise you will subscribe to the LANSA Image in the AWS Marketplace so that the LANSA Stack may use that image. This image is called an Amazon Machine Image or AMI.

Note 1: This step is not required when using the LANSA Product Centre AWS account.

Before You Begin

Obtain access to the AWS console to use AWS Marketplace

Step 1. Locate the LANSA Scalable AMI

- 1. Go to https://aws.amazon.com/marketplace
- 2. In the search box type "LANSA Scalable License" and press Enter
- 3. Select the LANSA Scalable License. The LANSA Scalable License AMI details are displayed
- 4. Browse the page to ensure it's suitable for your purposes and then click on the Continue button
- 5. Click on the Manual Launch tab.
- 6. Select the AMI version appropriate for your LANSA Application version.
- 7. Click on the Accept Terms button
- 8. Within a few moments you will be subscribed

Summary

What You Should Know

- How to subscribe to the LANSA image.
- This is only performed once for a specific Marketplace image. You will be able to use the image in any supported AWS region.

CTE030 – Upload your LANSA WAM Application

Objective

In this exercise you will upload your LANSA WAM Application MSI to AWS BLOB storage – S3. If you are only interested in creating a stack with the demonstration application, this step is not required.

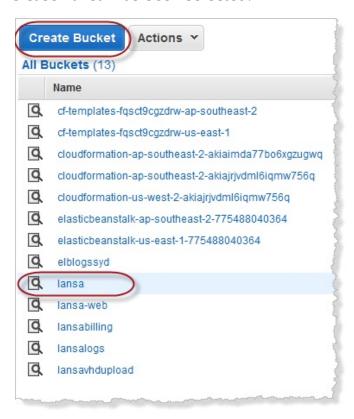
Before You Begin

You must have access to the LANSA Image (AMI)

You must have constructed your LANSA MSI.

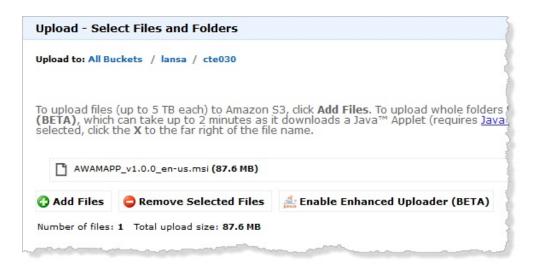
Step 1.Create a bucket

- 1. In the AWS Console window, select S3
- 2. Create a Bucket to store your MSI or select an existing bucket. If you create a new bucket use all lower case letters as some uses of S3 names are case sensitive. In this case 'lansa' has been selected.

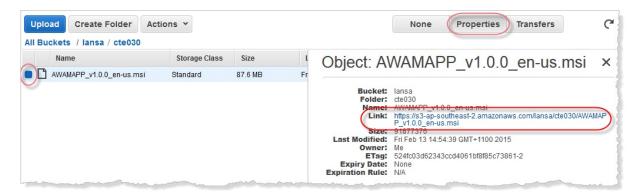


- 3. Create a Folder in your bucket or select an existing folder. 'cte030'
- 4. Select the MSI to upload to S3.
 - a. Click the *Upload* button
 - b. Click Add Files
 - c. Navigate to the directory where the LANSA MSI has been created
 - d. Select the MSI and click Open.

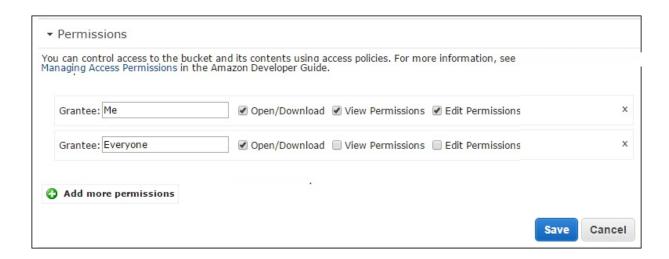
Your dialog will look something like this:



- 5. Click Set Details and then Set Permissions.
- 6. Check Make Everything Public or assign your own custom permissions.
- 7. Click Start Upload
- 8. When it's finished uploading, select the MSI and click the *Properties* tab on the right hand side to show the URL (Link) of the MSI in S3. Copy the URL and save it in a temporary document so you can paste it into the Template in the next exercise.



Once MSI has been uploaded, click the 'Properties tab on the right hand side under *Permissions* and set the options as shown.



Summary

What You Should Know

• How to upload your LANSA MSI to AWS S3 BLOB storage and retrieve its URL so you may specify it when creating the LANSA Stack.

CTE040 – Create the LANSA Stack

Objective

In this exercise you will use the AMI that you authorised in exercise CTE015 to instantiate a LANSA Stack.

Before You Begin

You must have completed the following exercises:

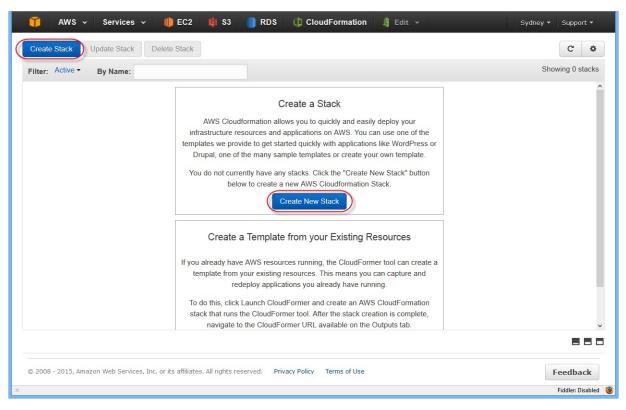
CTE015 – Subscribe to the LANSA Image (if you are not already permitted to use the LANSA Image)

CTE030 – Upload your LANSA WAM Application (if you want to deploy your own application. The demonstration does not require this step)

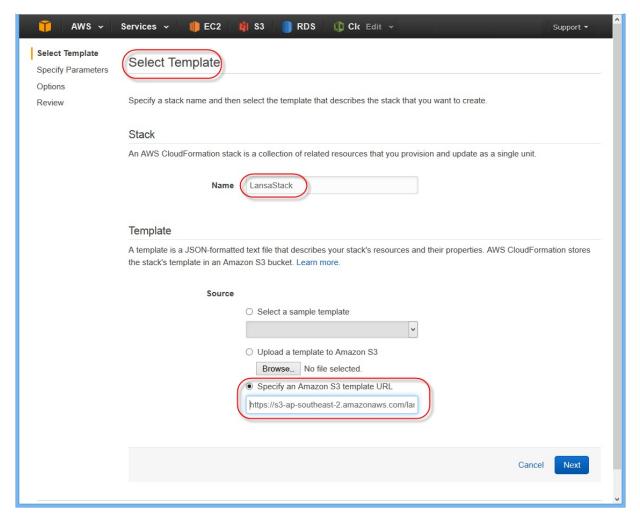
Step 1. Select Template

A CloudFormation stack requires a template. You will be using the template: https://s3-ap-southeast-2.amazonaws.com/lansa/templates/lansa-master-win.cfn.template. Copy this URL ready to paste it into the *Select Template* dialog.

1. In the *AWS Console CloudFormation window*, select *Create Stack* or *Create New Stack*. Create New Stack is only displayed if you do not currently have any stacks.



The Select Template dialog opens:



- 2. Specify the name of the stack e.g. LansaStack, select *Specify an Amazon S3 template URL*, and then paste in the template URL that you have saved: https://s3-ap-southeast-2.amazonaws.com/lansa/templates/lansa-master-win.cfn.template.
- 3. Click *Next* to go to the next step

Step 2. Specify Parameters

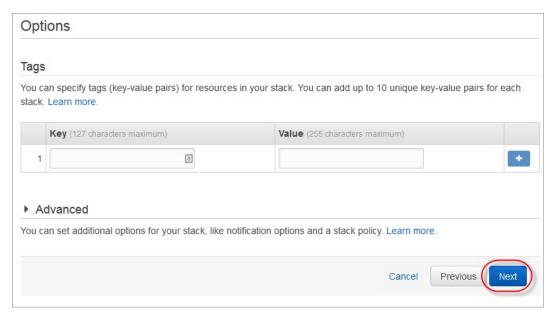
Specifying the parameters will define how your LANSA Stack behaves. It is the most important step in creating a stack and needs to be considered carefully. But, the worst that may happen is you need to recreate the stack again. You will see that instantiating a stack is not difficult to do, but understanding the effects of the various parameters and what will suit your application is another matter. Still, it's a great environment to try it and see what happens.

The user interface provided by AWS for CloudFormation templates is very basic. The parameters are in alphabetical order in a list. The most important parameters are listed at the top by prefixing the parameter name with a number to ensure it displays in the right order. All the help text for this dialog is provided on the dialog.

- 1. The most important parameters that you must specify are (in template order):
 - a. LansaMSI, unless you are running the demo
 - b. DBPassword
 - c. WebPassword
 - d. KeyName
 - e. RemoteAccessLocation
 - f. WebPassword
- g. It's also advisable to change the DBUsername and WebUser to non-default values
- 2. The description of some of the parameters starts "Update Stack:". These parameters are only relevant when updating the stack. All the Trigger parameters are examples of this and they have no effect when creating the stack, so you may safely skip them for the moment. Exercises using the Triggers are provided later on.
- 3. Review each of the parameters and its help text in the *Specify Parameters* dialog
- 4. Click Next.

Step 3. Options & Review

1. On the Options dialog click *Next*



- 2. On the *Review* dialog, scroll down to the bottom and acknowledge the Capabilities check box and click *Create*.
- 3. The stack will now be created. This process may take about an hour.

Step 4. Test the Stack

- 1. When the status of the stack becomes *Create Complete*, it will be almost ready to run.
 - Click the *Refresh* button to make sure the status messages are up to date. Give the web server another 10 minutes to setup the LANSA application.
- 2. Click on the *Stack Name* checkbox and the *Outputs* tab and then copy the LANSA web site URL.
- 3. Open a browser and paste in the URL. For the demonstration, append this to execute the WAM application: /cgi-bin/lansaweb? wam=DEPTABWA&webrtn=BuildFirst&ml=LANSA:XHTML&part=DEX&
- 4. A WAM should be displayed.

Summary

What You Should Know

• How to instantiate a LANSA stack in AWS.

CTE050 – Watch the LANSA Stack Scale In

Objective

In this exercise you will change the settings manually on the Auto Scaling Group to force the LANSA Stack to Scale Out 2 instances and then watch it automatically scale back in again.

Before You Begin

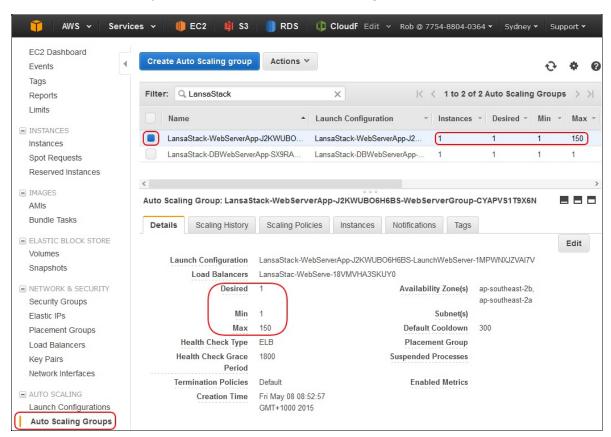
You must have completed the following exercises:

CTE040 – Create the LANSA Stack

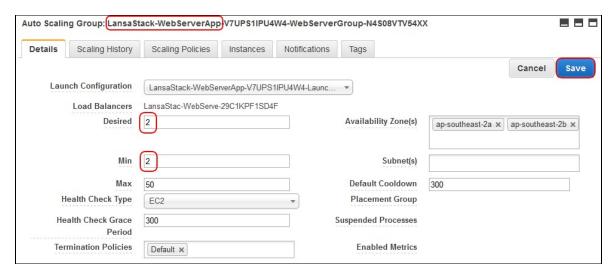
Step 1. Select Auto Scaling Group

- 1. In the AWS Console select the EC2 page and the Auto Scaling Group down the bottom of the left hand column.
- 2. At least two scaling groups should be displayed: one name starts *LansaStack*-**DB**WebServer... and the other starts with *LansaStack*-*WebServer*... Select the latter Auto Scaling Group. The *Details* tab should be displayed below the list. If not, click on one of the icons circled in green and ensure the *Details* tab is selected. Notice the *Instances*, *Desired* and *Min* values are 1 both in the list and the *Details*, and the *Max* is 150.

Note: Do not change *LansaStack***DB***WebServer* settings. It's important that there is 1 and only 1 of these instances running.



3. Click Edit to change these settings. Set the Desired to 2 and the Min to 2. This will instantiate an extra Web Server instance.



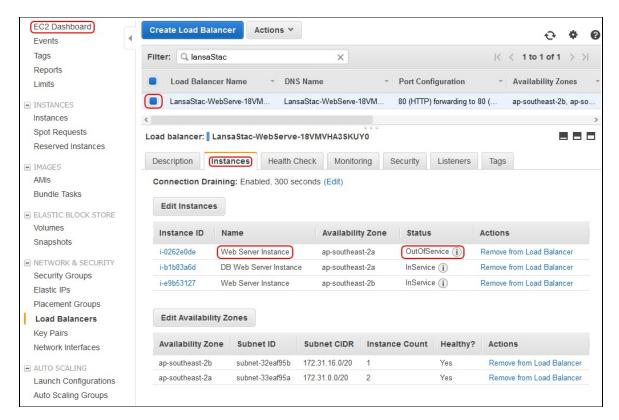
4. Click Save to apply the new settings. Notice the list is updated with the new settings and the Instance count is 1. Within a short time that will change to 2. Click the refresh icon to see the change.



5. You may see this instance initializing in the EC2 Instances view. You have forced the Auto Scaling Group to create 1 new instance.

Step 2. Wait for the Auto Scaling Group to Scale Out

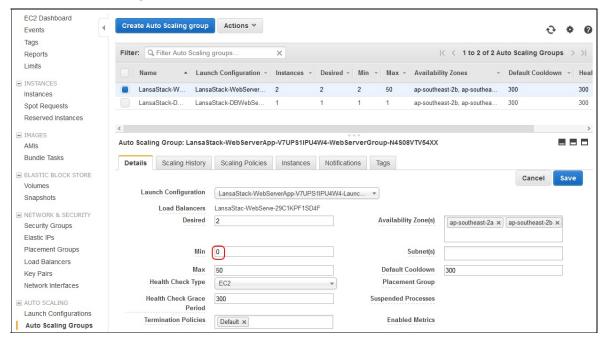
1. You must wait for the instance to initialize before configuring the stack to Scale In. You may determine when the instances are ready by looking at the state of the Load Balancer. Navigate to the EC2 Load Balancers window and select the LansaStack Load Balancer and the Instances tab. You will see that the Web Server instance is OutOfService. Until it is InService, the Load Balancer will not use it. It takes something like 20 minutes to fully initialize the demonstration application. Your own application may vary greatly from this.



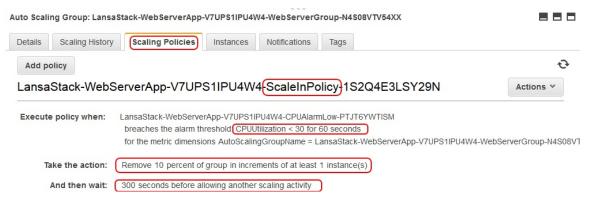
2. When all 3 instances have a Status of In Service then you are ready for the next step.

Step 3. Allow the Auto Scaling Group to Scale In

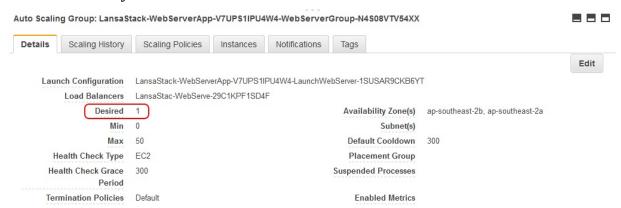
- 1. Navigate back to the Auto Scaling Group window. Select the Auto Scaling Group LansaStack-WebServer, click on the Details tab and click Edit. This is just the same as you did for Step 1.
- 2. Change Min to 0. This will allow the Auto Scaling to reduce the number of running instances (Desired setting) below 2 if the Scaling Policy is satisfied. Save the change.



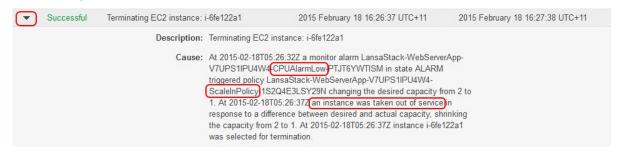
3. Click on the Scaling Policies tab. The policy names are quite lengthy. In the centre of the name you will see a policy named ScaleInPolicy. This policy is set to execute when the CPU Utilization drops below 30% for 60 seconds. When that occurs, I instance will be removed from the Auto Scaling Group. Then there will be a pause of 5 minutes before another scaling activity will be allowed. This is to stop thrashing of instances constantly starting and stopping. In fact in a production system the execution policy is more likely to average over a longer period so that dips of inactivity are averaged out.



4. If you click refresh you will probably see that the Desired setting has been automatically decremented to 1.



5. A scaling activity has occurred. Click on the Scaling History tab and expand the message using the arrow on the left. You can see the cause of this scaling activity.



- 6. 5 minutes later another scaling activity will occur and reduce the Desired number of instances to 0, triggering the last instance in this Auto Scaling Group to be removed from service.
- 7. You may peruse the other scaling activities that have occurred and see your user request to force the creation of the 2 web server instances to Scale Out the ASG and then the 2 automatic scaling events to Scale In the ASG.

Summary

What You Should Know

• How to manually scale a stack and how AWS performs scaling automatically based on Cloud metrics you specify.

CTE060 – Apply Windows Updates to all Web Server Instances Objective

In this exercise you will trigger the application of Windows Updates to all the Web Server instances. As it's an exercise you may do this at any time, but with a production system you would schedule a time to perform this maintenance activity and take the application offline and perform tests to ensure your application is still working as expected.

N.B. The AWS Console status will change to UPDATE_COMPLETE when the FIRST part of the process is complete. That is, it's updated the definition of the stack. The actual action you have requested is not started until the next time the agent on the EC2 instance checks for changes. This occurs at 1 minute intervals. The action must then be executed. For Windows Updates in particular this may take a long time. If you RDP into the instance before it's complete it may effect whether an update is installed successfully or not. If a reboot is required your RDP session will be ended.

Also, only apply one trigger at a time and allow sufficient time for the action to complete before applying another trigger.

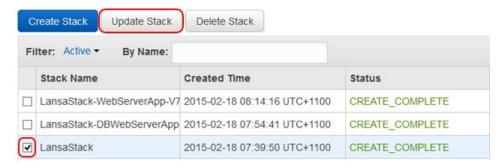
Before You Begin

You must have completed the following exercises:

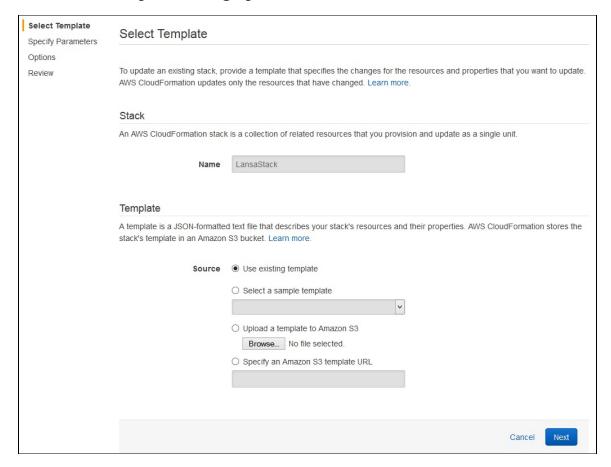
CTE040 – Create the LANSA Stack

Step 1. Select Stack to Update

- 1. In the *AWS Console CloudFormation window*, you will see three stacks listed. Click the stack that bears just the name of the stack which you created in exercise CTE040. E.g. LansaStack.
- 2. Then click Update Stack:



3. The *Select Template* dialog opens:

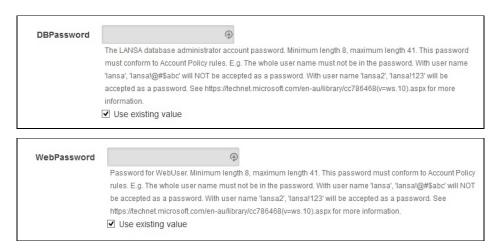


4. Just click *Next* to go to the next step

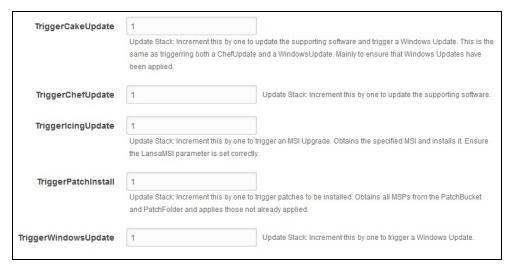
Select Template Specify Parameters	Specify Parameters		
Options Review	Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template. Parameters		
		The maximum size of the database. Must be between 20 and 1024 GB. Maximum 20 for 'sqlserver-ex'. Minimum 200 for 'sqlserver-ee' and 'sqlserver-se'.	
		DBEngine	sg/server-web v

Step 2. Specify Parameters

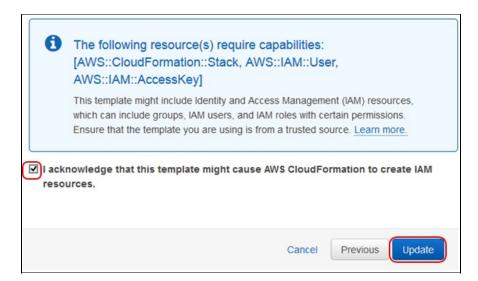
1. Click the Use Existing Value checkbox on the two parameters which use it: DBPassword and WebPassword. You must do this otherwise the Stack update will fail.



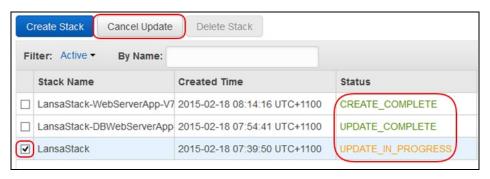
2. Scroll down the list of parameters to the *TriggerWindowsUpdate* parameter.

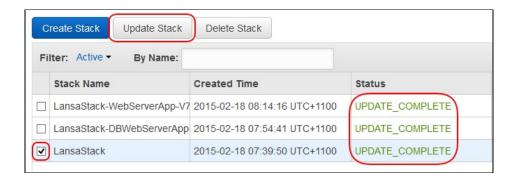


- 2. Increment the value by 1. In this example, enter 2 and click Next.
- 3. On the Options dialog click Next
- 4. On the Review dialog, check the 2 password parameters display ellipsis and the *TriggerWindowsUpdate* parameter is the value you expect.
- 5. Scroll to the end of the Review dialog and acknowledge the capabilities, then click Update:



6. The list of stacks is displayed and the stack you updated will have a status of UPDATE_IN_PROGRESS. Within a short time all 3 stacks will show UPDATE_COMPLETE. The Windows Update itself is not complete, just the application of the change to the template.





Step 3. Check that Windows Updates have been Applied

1. To perform this step you will need to know how to obtain RDP access to the web servers and check the Windows Updates status on the machine.

Summary

What You Should Know

• How to apply Windows Updates to all the running Web Servers by updating the LANSA Stack.

CTE070 – Apply LANSA Patch to all Web Server Instances Objective

In this exercise you will trigger the application of LANSA Patches to all the Web Server instances. As it's an exercise you may do this at any time, but with a production system you would schedule a time to perform this maintenance activity and take the application offline and perform tests to ensure your application is still working as expected.

N.B. The AWS Console status will change to UPDATE_COMPLETE when the FIRST part of the process is complete. That is, it's updated the definition of the stack. The actual action you have requested is not started until the next time the agent on the EC2 instance checks for changes. This occurs at 1 minute intervals. The action must then be executed. How long the action takes to execute is dependent on what you have asked it to do. It may take many minutes.

Also, only apply one trigger at a time and allow sufficient time for the action to complete before applying another trigger.

Before You Begin

You must have completed the following exercises:

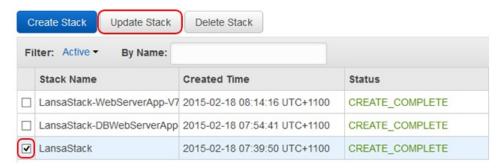
CTE040 – Create the LANSA Stack

CTE030 – Upload your LANSA WAM Application (but to upload your patches)

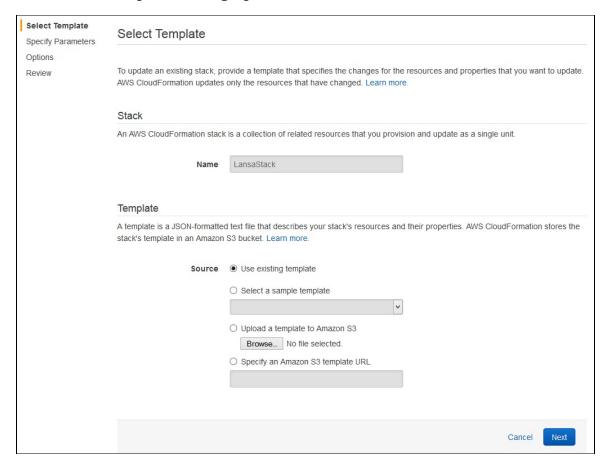
The patches for your application must have been uploaded already. You can follow the same procedure as explained in CTE030 – Upload your LANSA WAM Application but instead of uploading the MSI, upload the patches. The same S3 folder should be used as was used for the MSI.

Step 1. Select Stack to Update

- 1. In the *AWS Console CloudFormation* window, you will see three stacks listed. Click the stack that bears just the name of the stack which you created in exercise CTE040. E.g. *LansaStack*.
- 2. Then click Update Stack.



The Select Template dialog opens:

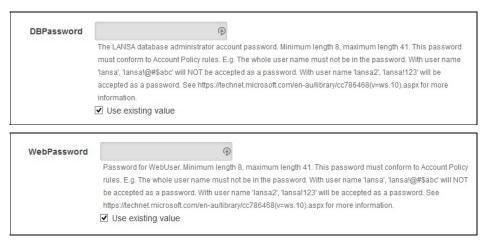


3. Just click *Next* to go to the next step

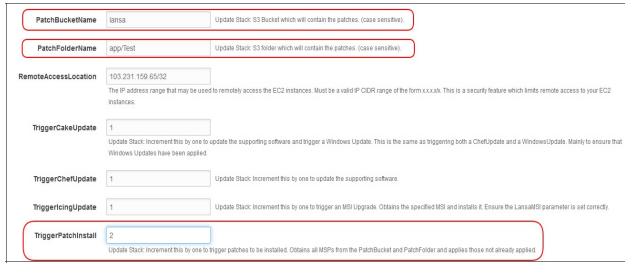
Select Template Specify Parameters	Specify Parameters		
Options Review	Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template. Parameters		
		The maximum size of the database. Must be between 20 and 1024 GB. Maximum 20 for 'sqlserver-ex'. Minimum 200 for 'sqlserver-ee' and 'sqlserver-se'.	
		DBEngine	sg/server-web v

Step 2. Specify Parameters

1. Click the Use Existing Value checkbox on the two parameters which use it: *DBPassword* and *WebPassword*. You must do this otherwise the Stack update will fail.



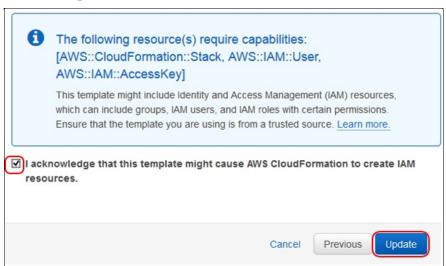
2. Scroll down the list of parameters to the *PatchBucketName*, *PatchFolderName* and *TriggerPatchInstall* parameters.



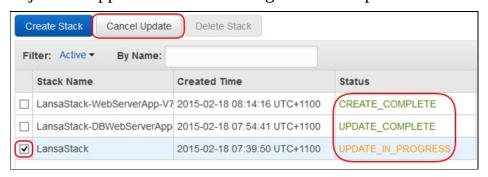
- 3. Ensure the PatchBucketName and PatchFolderName are correct. In particular the PatchFolderName may be just the folder where the patches are located, or a part of the patch name to restrict the installation to a subset of the patches in the directory. Foer example to install just one patch specify the entire name "/app/test-patches/AWAMAPP_v1.0.0.2_en-us.msp". Or all the patches for version 1.0.0 "/app/test-patches/AWAMAPP_v1.0.0." (N.B. include the last '.'.
- 4. Increment the *TriggerPatchInstall* parameter by 1. In this example, enter 2

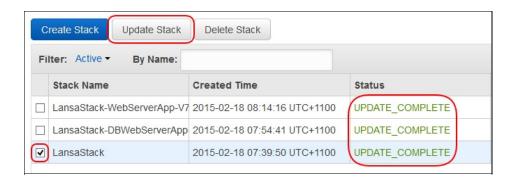
and click Next.

- 5. On the Options dialog click Next
- 6. On the *Review* dialog, check the 2 password parameters display ellipsis and the *PatchBucketName*, *PatchFolderName* and *TriggerPatchInstall* parameters are the values you expect.
- 7. Scroll to the end of the *Review* dialog and acknowledge the capabilities, then click *Update*:



8. The list of stacks is displayed and the stack you updated will have a status of *UPDATE_IN_PROGRESS*. Within a short time all 3 stacks will show *UPDATE_COMPLETE*. The LANSA Patch itself may not have completed, just the application of the change to the template.





Step 3. Check that the patch has been Applied

1. To perform this step you will need to know how to obtain RDP access to the web servers and check the LANSA Application installation status on the machine.

Summary

What You Should Know

• How to apply LANSA Patches to all the running Web Servers by updating the LANSA Stack.

CTE080 – Apply LANSA Application Upgrade to all Web Server Instances

Objective

In this exercise you will trigger the application of a LANSA Upgrade to all the Web Server instances. As it's an exercise you may do this at any time, but with a production system you would schedule a time to perform this maintenance activity and take the application offline and perform tests to ensure your application is still working as expected.

N.B. The AWS Console status will change to UPDATE_COMPLETE when the FIRST part of the process is complete. That is, it's updated the definition of the stack. The actual action you have requested is not started until the next time the agent on the EC2 instance checks for changes. This occurs at 1 minute intervals. The action must then be executed. How long the action takes to execute is dependent on what you have asked it to do. It may take many minutes.

Also, only apply one trigger at a time and allow sufficient time for the action to complete before applying another trigger.

Before You Begin

You must have completed the following exercises:

CTE040 – Create the LANSA Stack

CTE030 – Upload your LANSA WAM Application (but to upload your new MSI)

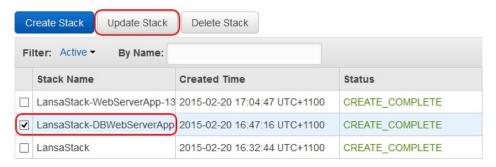
The upgrade for your application must have been uploaded already. You can follow the same procedure as explained in CTE030 – Upload your LANSA WAM Application. The same S3 folder should be used as was used for the first installation.

It's important to understand that the *DBWebServer* instance must be fully upgraded before attempting to update the *WebServer* stack. This is because all database changes must be applied only by *DBWebServer* and must be completed before the *WebServer* stack attempts to use those changes.

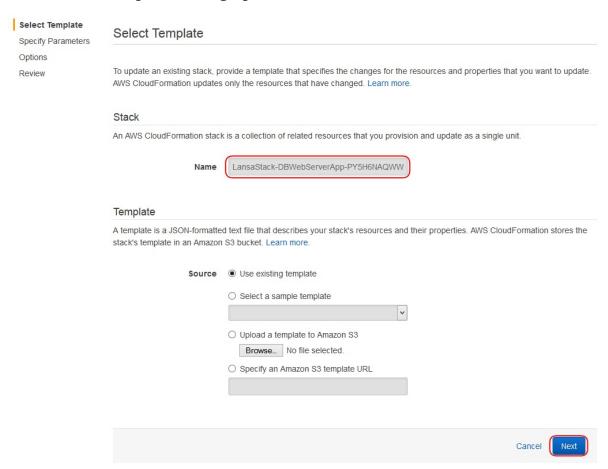
Hence the two nested stacks are updated rather than the parent stack.

Step 1. Select Stack to Update

- 1. In the *AWS Console CloudFormation* window, you will see three stacks listed. Click the stack that bears the name of the stack which you created in exercise CTE040. E.g. *LansaStack* with '–DBWebServerApp' appended.
- 2. Then click Update Stack.



The *Select Template* dialog opens:

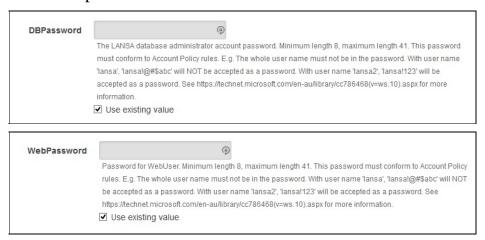


3. Just click *Next* to go to the next step

Select Template Specify Parameters	Specify Parameters		
Options Review	Specify values or use the default values for the parameters that are associated with your AWS CloudFormation template. Parameters		
		The maximum size of the database. Must be between 20 and 1024 GB. Maximum 20 for 'sqlserver-ex'. Minimum 200 for 'sqlserver-ee' and 'sqlserver-se'.	
		DBEngine	sg/server-web v

Step 2. Specify Parameters

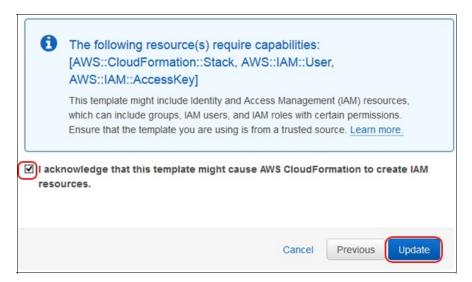
1. Click the Use Existing Value checkbox on the three parameters which use it: *DBPassword*, *DBName* and *WebPassword*. You must do this otherwise the Stack update will fail.



2. Scroll down the list of parameters to the *LansaMSI* and *TriggerIcingUpdate* parameters.



- 3. Specify the URL of your application upgrade and increment the *TriggerIcingUpdate* value by 1. In this example, enter 2 and click *Next*.
- 4. On the Options dialog click Next
- 5. On the *Review* dialog, check the 3 password parameters display ellipsis and the *LansaMSI TriggerIcingUpdate* parameters are the values you expect.
- 6. Scroll to the end of the *Review* dialog and acknowledge the capabilities, then click *Update*:



7. The list of stacks is displayed and the stack you updated will have a status of *UPDATE_IN_PROGRESS*. Within a short time the stack will show *UPDATE_COMPLETE*. The LANSA Upgrade itself has not completed, just the application of the change to the template.



Step 3. Check that DBWebServer has been Upgraded

- 1. To perform this step you will need to know how to obtain RDP access to the web servers and check the LANSA Application installation status on the machine.
- 2. Even if you do not RDP into the instance to check the installation status, it is highly recommended to check the CPU utilization of the EC2 instance and ensure it's returned to near zero. This ensures that the database upgrade has completed. Only then may you continue with the process.

Step 4. Upgrade the WebServer Stack

- 1. Once you are certain that the DBWebServer upgrade has completed, then repeat steps 1 to 3 for the *WebServer Stack*, whether or not you have instances currently running in that stack. All new instances will install the MSI last set in the template. So it's important to set it even when the WebServer stack is empty of instances.
- 2. Even if you do not RDP into the instance to check the installation status, it is highly recommended to check the CPU utilization of the EC2 instance and ensure it's returned to near zero. This ensures that the database upgrade has completed. Only then may you continue with the process.

Summary

What You Should Know

• How to upgrade a LANSA application in all the running Web Servers by updating the two nested LANSA Stacks.

Glossary

The following terminology is used in the Deployment Tool.

ActiveX External component

ActiveX External Components are ActiveX controls provided by a 3rd-party. Visual LANSA supports all standard ActiveX Interfaces.

To use an external ActiveX Control within the LANSA development environment, a corresponding LANSA ActiveX component is created. The external program information associated with the ActiveX control is identified within this ActiveX object.

The ActiveX control can then be used within Visual LANSA much the same as any other Visual LANSA component.

For more information, refer to ActiveX Controls in the Visual LANSA Developer's Guide.

ActiveX wrapped component

LANSA Components can be "wrapped" and made available to other vendors as an ActiveX Control.

The processing of "wrapping" the LANSA component requires the registration of the LANSA component and creation of a supporting session.cfg file in the environment where the ActiveX will be used.

For more information, refer to ActiveX Controls in the Visual LANSA Developer's Guide.

Application

"Application" has a dual meaning in the context of the Deployment Tool.

In the first instance an Application draws a boundary around the product you have developed using LANSA, encompassing all the bits and pieces which comprise the Application. This is a LANSA-developed Application.

In terms of the Deployment Tool an Application is a group of Packages. One or more Packages can be linked under the "umbrella" of an Application in the Deployment Tool. An Application is distributed for download, rather than a Package.

You may have more than one Application defined in the

Deployment Tool to deploy your LANSA-developed application. For example, if you need to deploy a client server application you will typically define a Deployment Tool Application to deploy the client side components and a separate Deployment Tool Application to deploy the server side components.

Application Server

For Just in Time deployment, the Application Server is the repository for all released deployment packages and has an active listener to receive requests from Client (or Target) PCs.

When the Application is launched on a Client PC this initiates communication with the Application Server to determine if updates to be downloaded. Any updates will be automatically applied on the Client PC before the Application is started.

Assembler

The person selecting the deployment tool options & settings described in this guide.

Base Language

The language to be translated from when performing LANSA Object or Parameter File translations based on a translation package.

The language must be defined in the partition where the package is installed.

Client PC

Client PC(s) are an integral part of a client-server installation. Typically the Client PC accesses a database on the server (rather than locally) and may rely on the server to perform some operations.

Files

Communication The communication files are Host Route Table (lroute.dat) and Listener Table (listener.dat). These files are maintained using the LANSA Communications Administrator interface.

Corporate **Application** Install

In a corporate environment it is assumed users do not have administrative rights to their PC therefore when installing an Application with a database an administrator of the machine is required to set up the shared database by executing an "All Users" install of the Application with Setup Database = TRUE (i.e. SUDB=1).

The end user can then install any subsequent updates which do not modify the database using a "Per User" install.

If an end user needs to be allowed to set up the database the Application deployment settings can be modified to ensure the database setup is allowed OR the installation can be executed from a command line using msiexec.exe and setting a parameter to change Setup Database to TRUE (i.e. SUDB= 1).

Refer to Personal Application Install for comparison.

Deployment Template

In the context of the Deployment Tool a Template is implements a predetermined set of values for the Package's settings and options. Objects can also be included in a Deployment Template.

By using Templates, you can reduce the number of options to be set each time you wish to create a package. You can create a purpose-built template or save a package as a template. A number of pre-recorded Templates are supplied with the

Deployment Tool.

Developer

The person who developed the Visual LANSA application being deployed. This person may also be the Assembler of the package.

DLL Upgrade

DLL upgrades are no longer supported.

The alternate approach is a Package Upgrade.

End User

The user of the Visual LANSA Application.

Globally Unique Identifier (GUID)

A Globally Unique Identifier, commonly called a GUID, is a unique reference number used to identify software.

GUIDs are usually stored as 128-bit values, and are commonly displayed as 32 hexadecimal digits with groups separated by hyphens, for example 5667BA66-1CC2-442E-8B80-03288BB8FDC0.

Installer

The person who installs the package on the end user's PC. This person may also be the end user.

Just in Time (JIT) Installation

Just-in-time package installation ensures the latest changes to a Package-developed application are installed before a user accesses the application. The Application Packages are defined with appropriate dependencies, for example install package A before package B, and then they are installed on a network server. When a user starts the application a check is performed to see if any upgrades needs to be installed before the application is launched. Any packages not previously installed on the client system are autoamtically installed according to the pre-defined dependencies before the application is launched.

Only installing changes when an application is used has the advantages of:

- Saving on set up time
- Ensuring consistent application versions on client systems.

MSI file

MSI file is an installer package file format used by Windows.MSI files are used for installation, storage, and removal of applications.

When you successfully build a Version, an MSI file is produced. This includes all the objects required to install your application whether it be a server installation, a client installation, a network client or other variation.

msiexec.exe

Msiexec.exe belongs to Windows Installer and is used to install, modify or configure applications from the command line which are installed from an MSI file.

MSP file

An application that has been installed using the Microsoft Windows Installer can be upgraded by reinstalling an updated installation package (a MSI file), or by applying a Windows Installer patch (an MSP file) to the application.

An MSP file contains updates to an application and describes which versions of the application can receive the patch.

Servicing applications by delivering a Windows Installer patch, rather than a complete installation package for the updated product can have advantages. A patch can contain an entire file or only the file bits necessary to update part of the file. This enables the user to download an upgrade patch that

is much smaller than the installation package for the entire product. It is not recommended that you include database changes as they are complex to manage and can easily result in a corrupted database.

Application

Network Client A Network Client application allows you to create packages that do not include LANSA executable objects. Typically, a package for a Network Client application will contain shortcuts to the server and indicate the target platform. It is intended for applications where the executables and runtime environment are installed on a central sever for shared use.

> A separate application must be defined to install the server side components, that is, the LANSA runtime environment, communications and compiled DLLs from functions and components.

Package

A package defines what and how your LANSA-developed Application will be deployed to a user's PC.

A Package is a generic term to refer to a Version or Patch.

Package Upgrade

There is only one type of Just-In-Time Upgrade, so this distinction - Package Upgrade - is superfluous. See Just-In-Time Installations.

Patch

A Patch is used to deliver a set of changes to a software product that has been installed using the Windows Installer. A software product can be upgraded by installing a new Version (MSI file) or by applying a Patch (MSP file).

A Patch is typically used to apply modified compiled objects, shortcuts and executables. A patch should not be used to deploy database changes.

A Patch is identified by its Patch Number. The Patch number has a direct relationship to the Version or Patch it was based on, for example Patch 1.0.0.1 would be the first patch against Version 1.0.0.

Personal Application Install

In the case of a Personal Application with a database the end user of the application is allowed to set up the database for an Application. To facilitate this the package must be built with the appropriate Database Setup options OR the

installation can be executed from a command line using msiexec.exe and setting a parameter to change the Setup Database Setup to TRUE (i.e. SUDB= 1).

Refer to Corporate Application Install for comparison.

Primary Installation

The main Application installation is generally referred to as the primary installation to distinguish it from the network client installation which is a small MSI file automatically created for most primary installations and delivered by the primary installation.

Target PC

The PC where the Visual LANSA application is to be deployed.

Template

See Deployment Template.

Translation Language

The language to be translated to when performing LANSA Object or Parameter File translations based on a translation package.

The language must be defined in the partition where the package is installed.

Translation Package

A translation package includes either (or both) of the *Deploy* with *Translation List* or *Translate Deployment Parameter File* options described in Settings .

When installed a translation package will create a desktop folder with two options:

- Install
- Translate Application

The Translate Application option will launch the Translate Object Details form supplied by LANSA.

The XTRANSLT template provides a basis for a translation package.

X_START

The X_START.SAV file stores any override values to be used in conjunction with an X_RUN.EXE command.

Typically, when a package is installed a desktop icon is created to launch the application. This icon has a LANSA X_RUN command associated with it. The X_START file is

used in conjunction with the X_RUN to prompt for values before launching the application and to provide useful default values to be used on the prompt dialog. This can introduce flexibility into how and what is launched when the application is started.

Version

A Version is a full install of the software product. A Version belongs to an Application.

A Version is identified by its Version Number. The Version Numbering is arbitrary but subsequent version numbers must be greater than previous ones.