http://www.lansa.com

LANSA

Newsletter
February 2010

LANSA V12

We are very proud to announce LANSA Version 12. It contains lots of new features and you will find all details it in this Newsletter.

The next issues will be discussed in this Newsletter:

- Windows Install and Upgrade.
- Easily Transfer Applications to Any Server.
- LANSA Enforcement Triggers.
- IBM i Client to a Windows Server.
- Unicode Support for Files.
- Use SQL Views and MySQL Database Tables.
- Create Tables via SQL.
- Long Userids and Passwords.
- SSL Encryption.
- Latest LANSA Integrator.
- Latest Framework and RAMP.
- Many Other Enhancements.

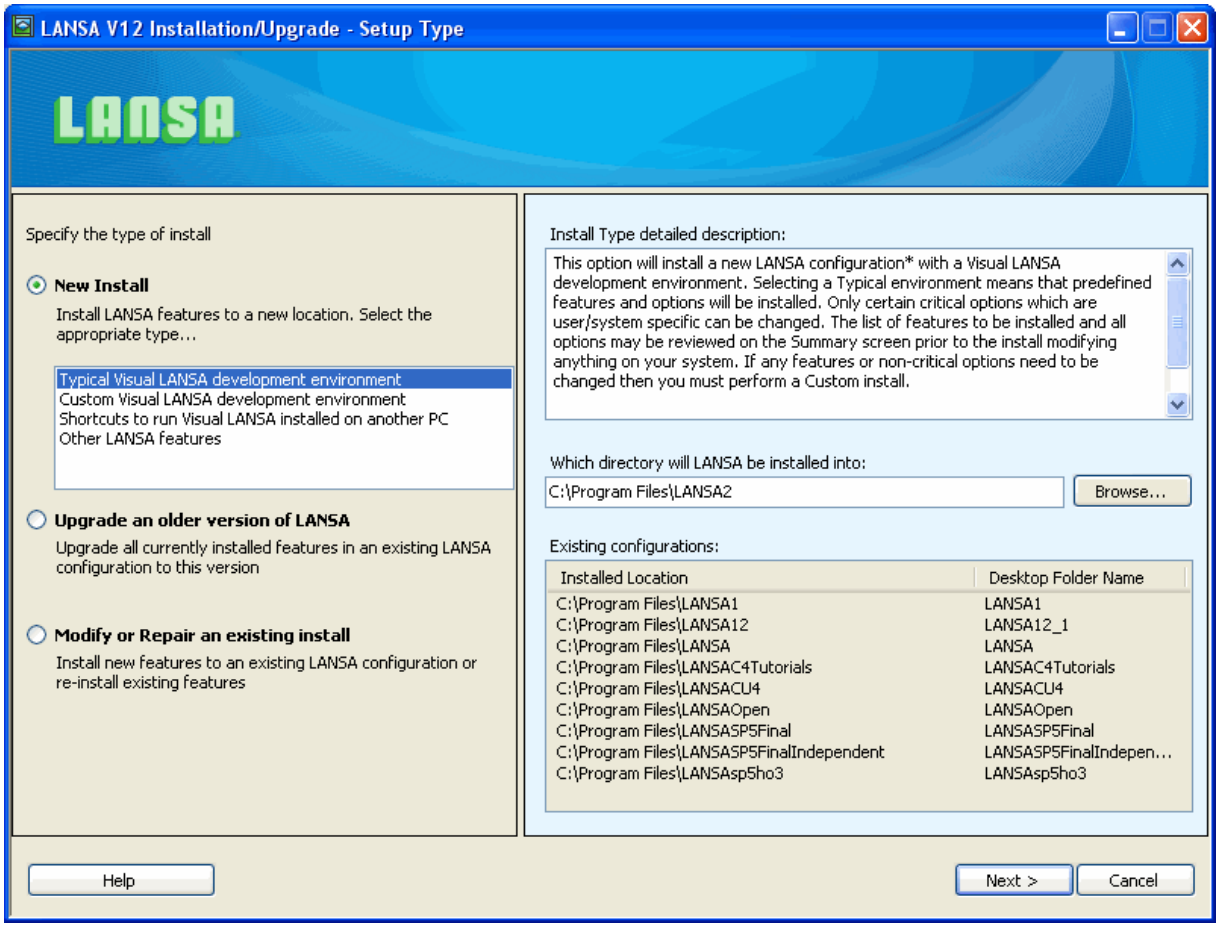
In This Issue

1. Windows Install and Upgrade

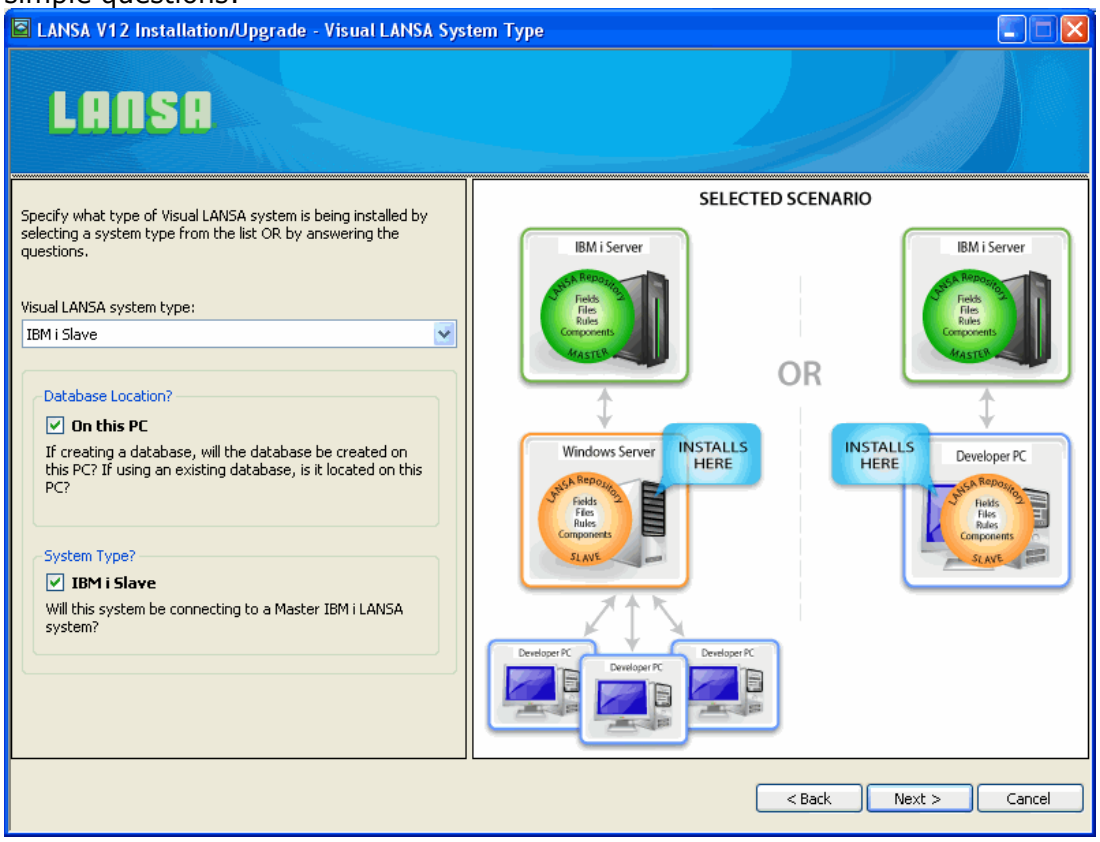
The V11 Installshield install has been replaced by a new .NET version. The Installation Wizard will walk you through the installation and upgrade process.

The new simplified and modernized LANSA Windows Install and Upgrade includes SQL Server 2008 Express as the Visual LANSA development database as well as components of the Visual Studio 2008 Express C++ compiler and runtime.

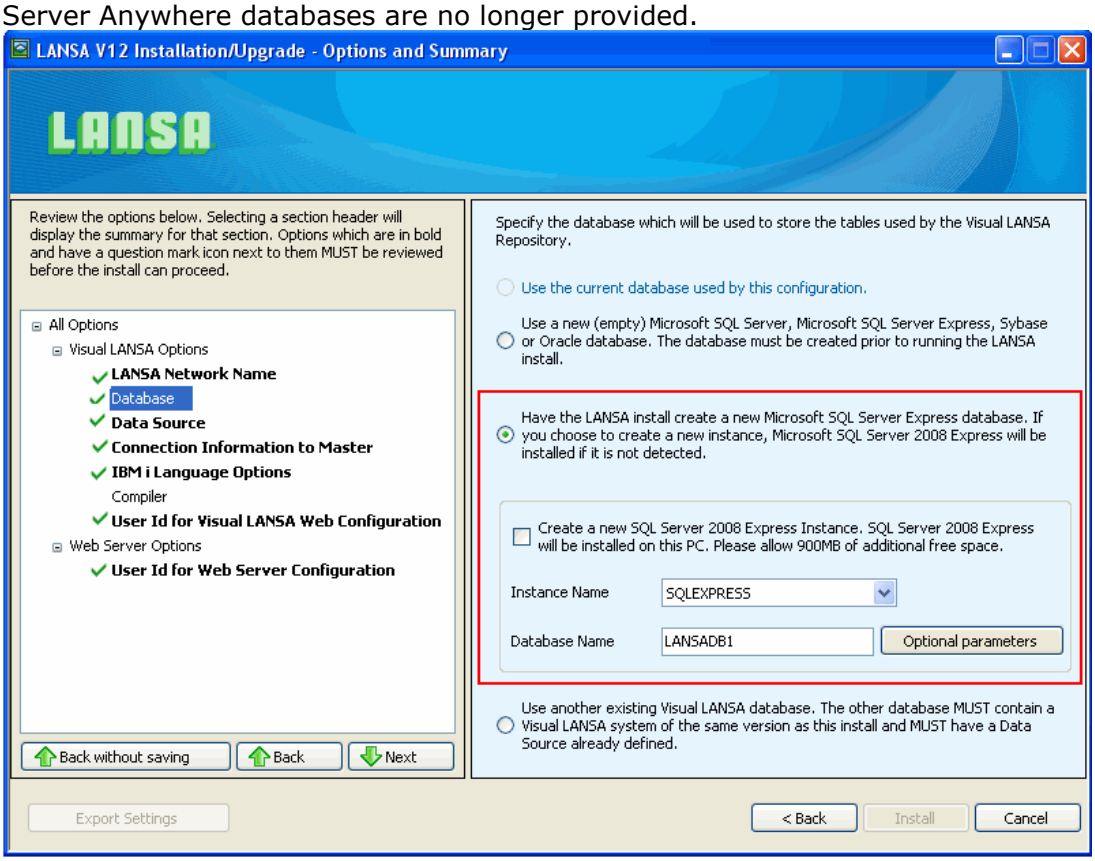
The installation user interface has been restructured for ease of use:



You can now easily determine the type of system you want to install by answering a few simple questions:



You have the option of installing SQL Server 2008 Express as the database. Adaptive Server Anywhere databases are no longer provided.

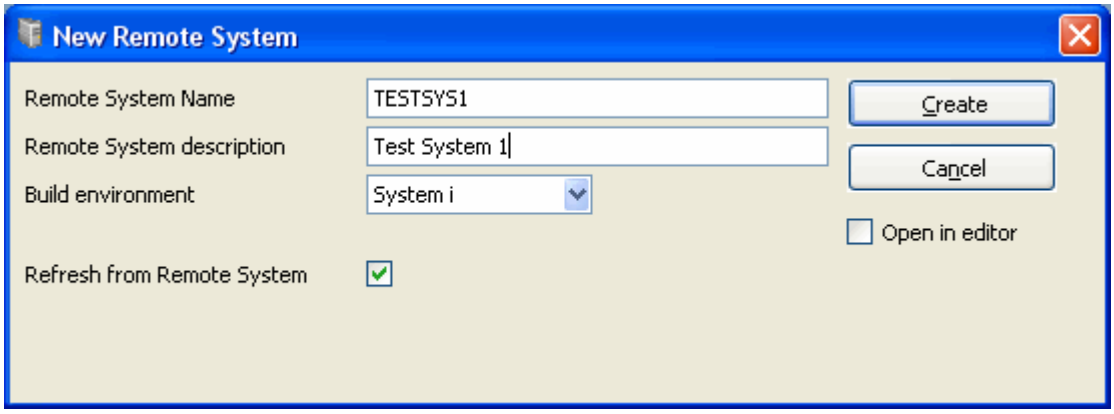


2. *Easily Transfer Applications to Any Server*

Allows you to develop entirely on a Windows Server and deliver code to either IBM i or Linux (and build the code). Windows would still be deployed with the Deployment Tool as a Compile is not required.
This allows you to keep your master source on a Windows server and deliver for systems testing.

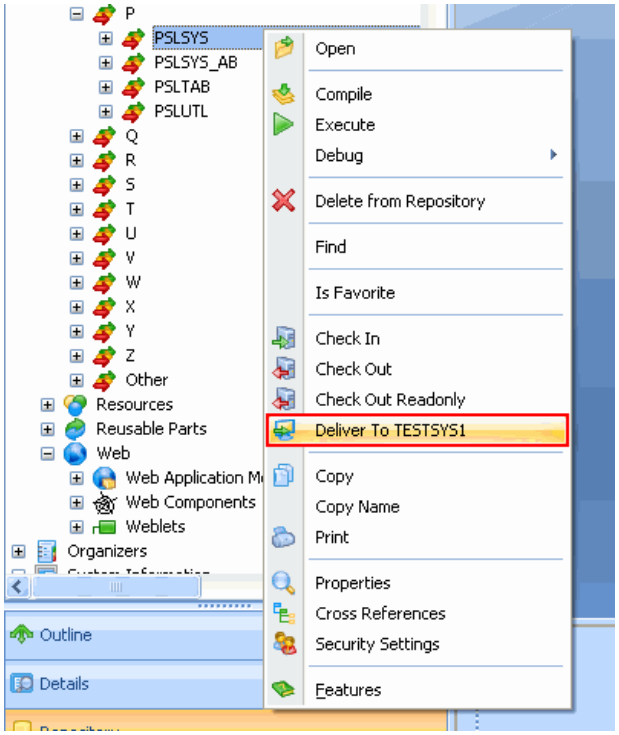
You can specify any server as remote server in your Visual LANSA environment and then easily deliver objects or entire applications to it. Unlike when checking objects into a master system, there is no validity checking of object status or task tracking.

This option is typically used when progressing from development to systems testing for a set of changes using a deployment master Visual LANSA environment. Refer to the Visual LANSA Administrators Guide for instructions on how to set up a Deployment System and Deliver objects to this environment. It is recommended to read the section "Restrictions and Assumptions" before proceeding.



The server can be any IBM I server which has a host route defined in the LANSA Communications Administrator.

When the remote server has been defined, you can use the Repository tab and right click on the objects to be transferred. The Deliver To option will be displayed in the pop-up menu. Editor Lists can also be selected to deliver the objects associated with the list definition.

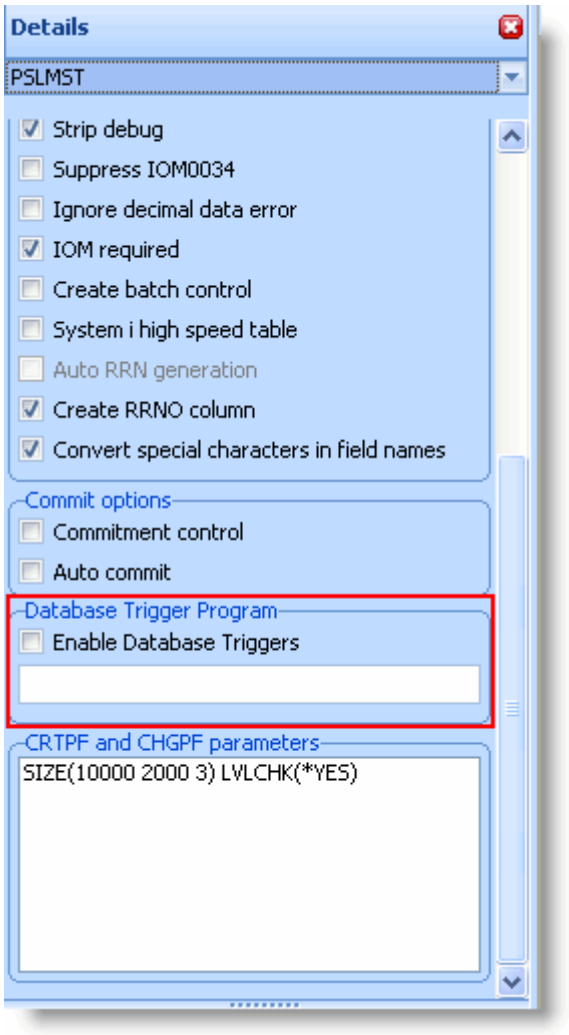


3. LANSA Enforcement Triggers

You can now enforce LANSA repository rules and triggers to be used by non-LANSA programs when they access a LANSA-defined file. This makes the LANSA repository rules available to all I/O done against a file.

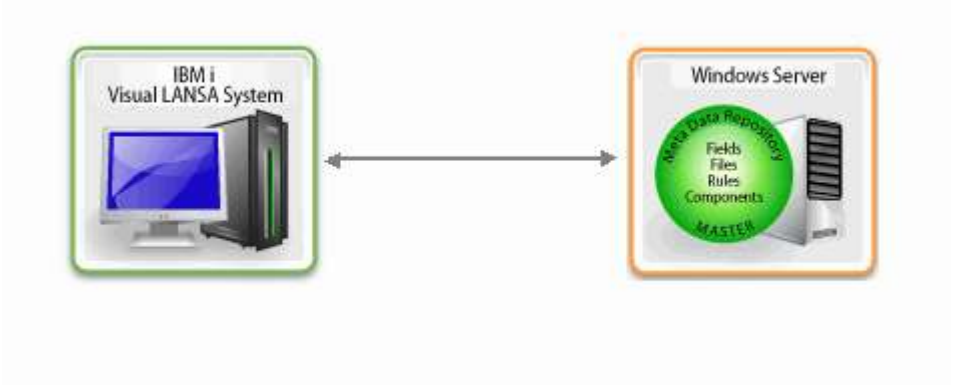
LANSA rules can be enforced for database I/O performed for example via 3GLs such as RPG and COBOL, via utilities such as DFU or STRSQL, or from off platform via ODBC or ADO.

LANSA enforcement triggers, called database triggers, are available for files in RDMLX partitions.



4. IBM i Client to a Windows server

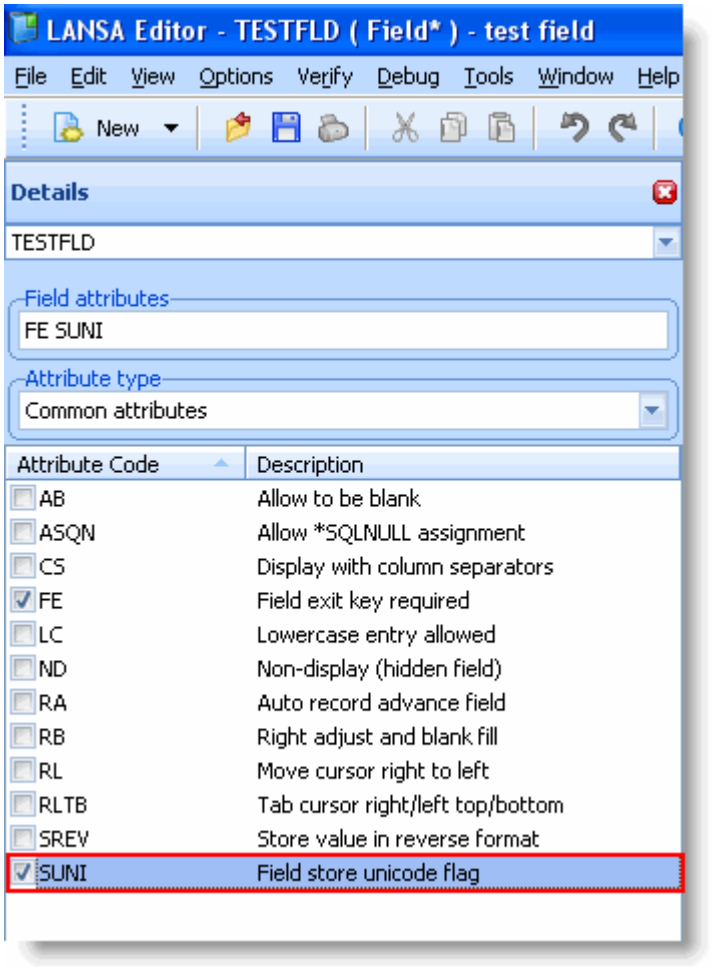
You may now establish a SuperServer connection from an IBM i Visual LANSA system to a Windows Server using the **DEFINE_ANY_SERVER** and **CONNECT_SERVER** Built-in Functions.
This allows IBM i applications to read and update databases on Windows servers.



5. *Unicode Support for Files*

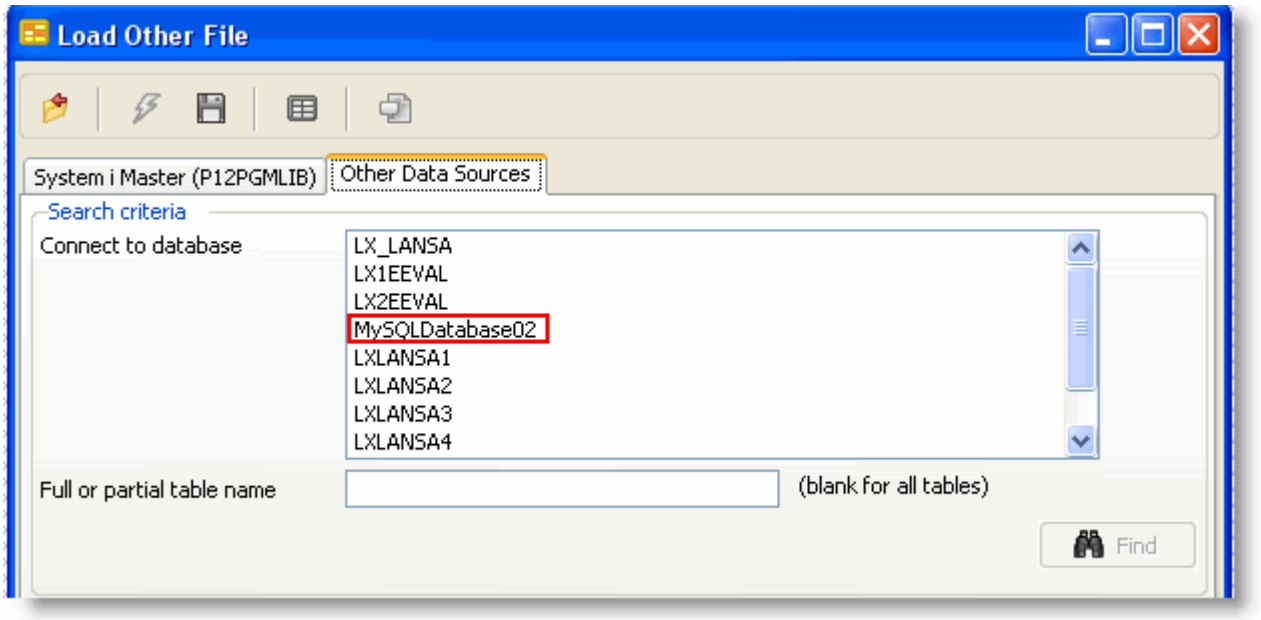
The attribute SUNI (Store in Unicode) is used to indicate that Char, String or CLOB fields are to be stored in **Unicode (UTF-16)** in the database.

The data is converted from the current code page to Unicode when written to the file and converted from Unicode to the current code page when read from the file. This means it is transparent to the RDML code that the field is Unicode in the database.



6. *Use SQL Views and MySQL Database Tables*

You can now load MySQL database tables as Other Files in RDMLX partitions.



You can also load definitions of SQL Views on both IBM i (RDMLX partitions) and Windows. When SQL Views are read-only (for example if defined over more than a single table), RDML database commands will be restricted to SELECT/FETCH commands.

7. Create Tables via SQL

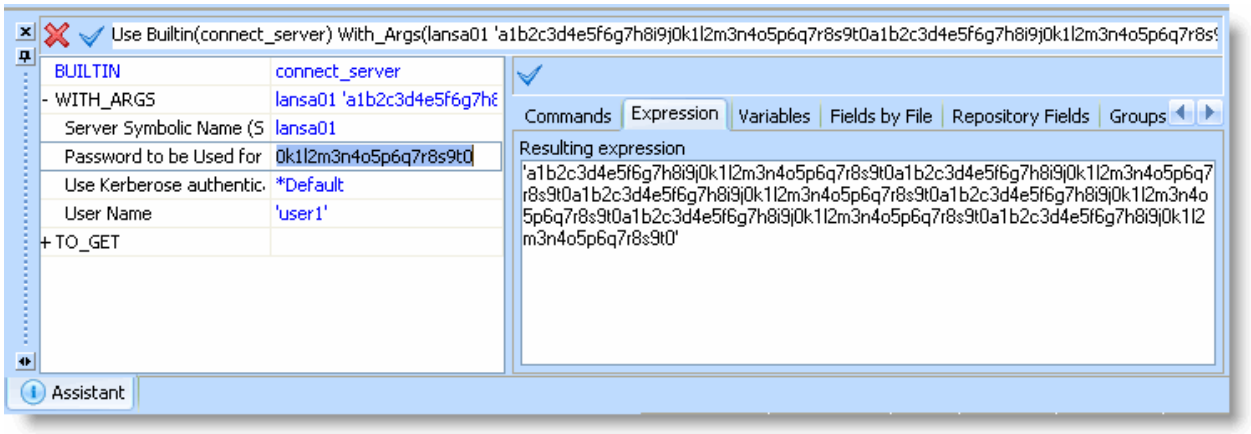
You can now create tables via SQL rather than DDS on IBM i in RDMLX partitions. IBM has indicated that the read performance of tables is improved when created by SQL as a larger read buffer is used.

When *SQL_BUILD is placed in data area DC@OSVEROP, and physical and logical files are recreated in an RDMLX partition, the physical file is created with SQL as a table, and logical views are still created with DDS, but as long as there is either no select/omit criteria or select/omit criteria and dynamic select Yes, a logical file will share its access path with an SQL index.

RDML files created in an RDMLX partition when *SQL_BUILD is in data area DC@OSVEROP can be exported to RDML partitions and re-exported to RDML partitions and will remain "SQL built" until they are rebuilt in an RDML partition.

8. Long Userids and Passwords

Long userids and passwords are now supported in many LANSA products. For example, the CONNECT_SERVER Built-in Function now supports 256 byte User Name and Password.



The products that support long userids and passwords include:

- Visual LANSA IDE
- The Host Monitor
- SuperServer to Windows and IBM i
- LANSA for the Web
- LANSA Open
- LANSA Open for .NET

The length of the userid and password is platform-specific and reflects the limitations of the platform. This includes case sensitivity when a platform is case sensitive.

Support based on platforms:

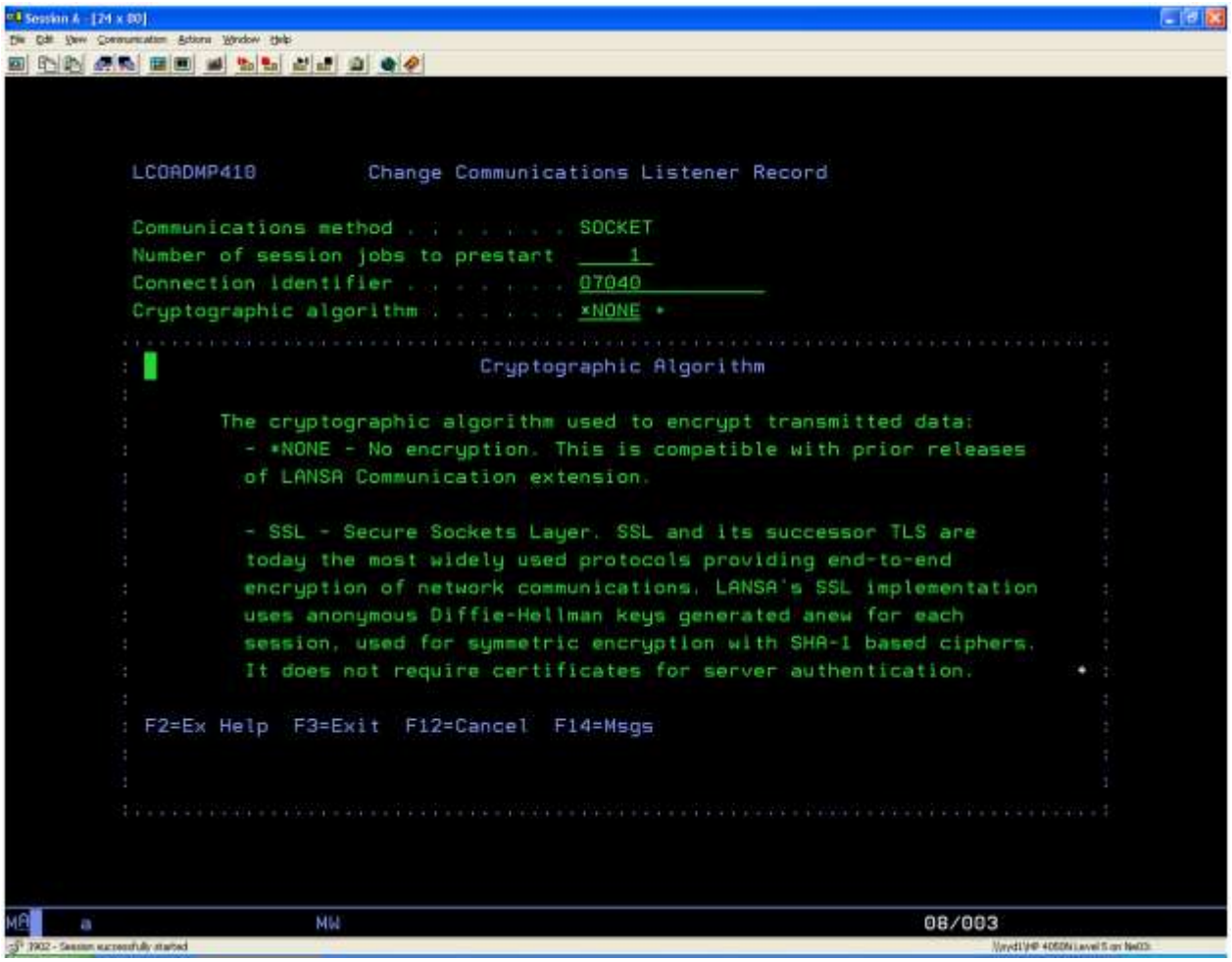
System	Length userid	Length password
IBM i	10	127
Windows	20	127
Linux	256	256

Note that the Visual LANSA IDE security is based on IBM i limitations and thus the userid is limited to 10 characters.

9. SSL Encryption

OpenSSL Encryption to IBM i for Client/Server communications is now available as a server-side option for secure encryption of network communications.

The handshake between server and client uses anonymous ephemeral Diffie-Hellman keys, with the following data transfers encrypted with a per-session symmetric key. The previous encryption algorithms (DES and Twofish) are still available, but we recommend using SSL instead. Please note that SSL does carry a processing overhead and may reduce transfer speeds.



10. Latest LANSA Integrator

Refer to the LANSA Integrator EPC835 documentation for information about the latest new features.

11. Latest Framework and RAMP

See the **Visual LANSA Framework guide** and the **RAMP guide** for information about the latest new features.

12. Many Other Enhancements

➢ Export/Import I/O Modules and OAMs without Files

IBM i export/import allows the export and import of a compiled I/O module and an OAM for a file without the physical or logical files being exported. The file definition will be updated from the export. This is helpful if you have just changed a rule and so only need to OAM to be updated.
Frameworks and Groups can now be omitted when components are exported.

➢ Free Format version of the SELECT_SQL command

The free format version of the SELECT_SQL command allows any SQL that is valid for a particular database engine to be used at execution time. No parsing is performed of the SQL either at compile time or runtime.

The entered SQL command is passed exactly as it is to the database engine. It is the responsibility of the programmer to ensure that the data returned by the database engine matches the list of fields in the FIELDS parameter.

This form of the SELECT_SQL command can only be used in RDMLX functions and components.

For example:

➢ Active-X Exception Handling Traps Errors at Runtime

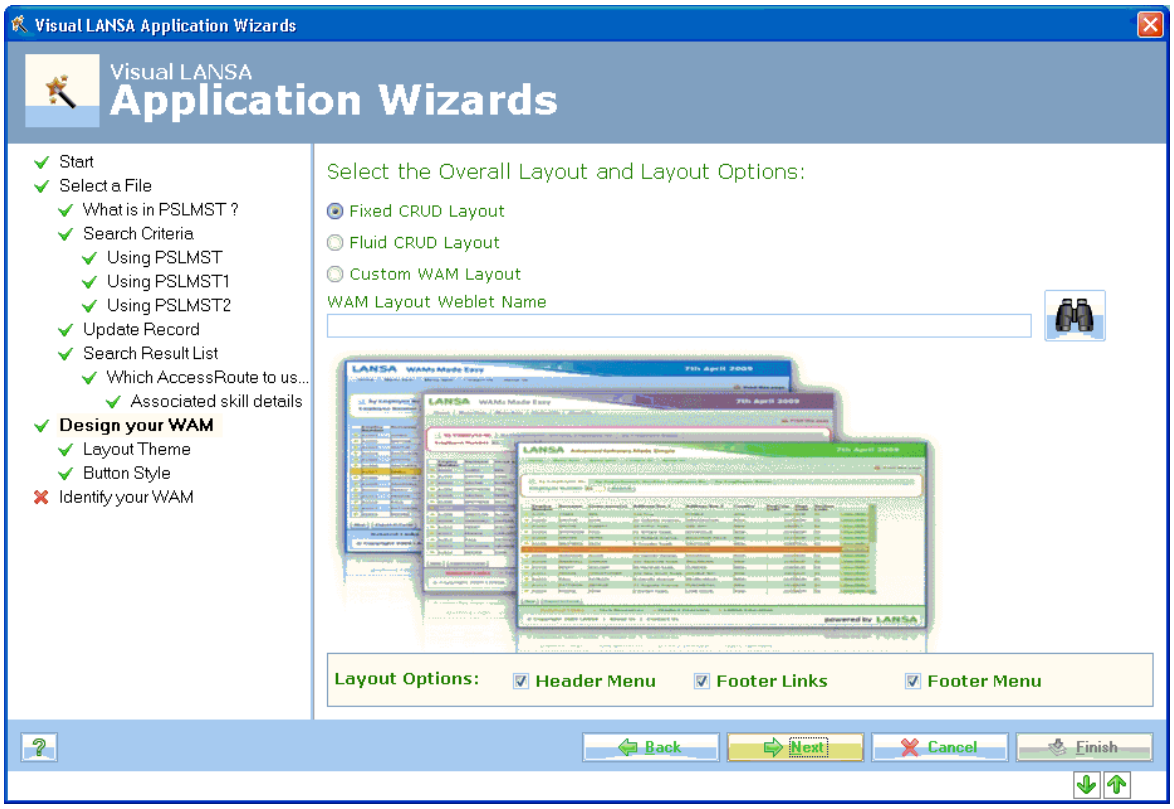
Active-X Exception Handling now traps failed HRESULT values in Visual LANSA at runtime so that they do not cause application failures. Access is provided to the HRESULT values.

Details are published in the ActiveX Controls section of the Visual LANSA Developer Guide.

➢ Wizard to Create Web CRUD Applications

Application Wizards are now available from the Tools menu of the Visual LANSA IDE.

The first available Wizard generates Web CRUD applications in RDMLX partitions. You answer a series of simple questions using the Wizard interface after which a complete Web Application Module (WAM) is generated, compiled and optionally executed.



➢ Support for LOBs in WAMs

MIME type and LOB content is now returned for WAM webroutines.

The output of any nominated content file with appropriate http headers is also supported. This is enabled via a **new weblet named std_lob** and a new Webroutine parameter Response and properties. The option of removing the file once sent is also available.

For example:

- New Response parameter & properties:

```
Webroutine Name(FETCH_REPORT_PDF)      Response(#http1)
Desc('BLOB serving WebRoutine')
#http1.ContentFile := #LUSERPTH + 'report.pdf'
#http1.RemoveFile := True
Endroutine
```

➢ SQL Statements are Cached for Re-execution

Reusable SQL statements are now cached for re-execution to improve performance. By default, up to 50 reusable SQL statements are cached for re-execution. The statements are automatically dropped from the cache when a file is closed.

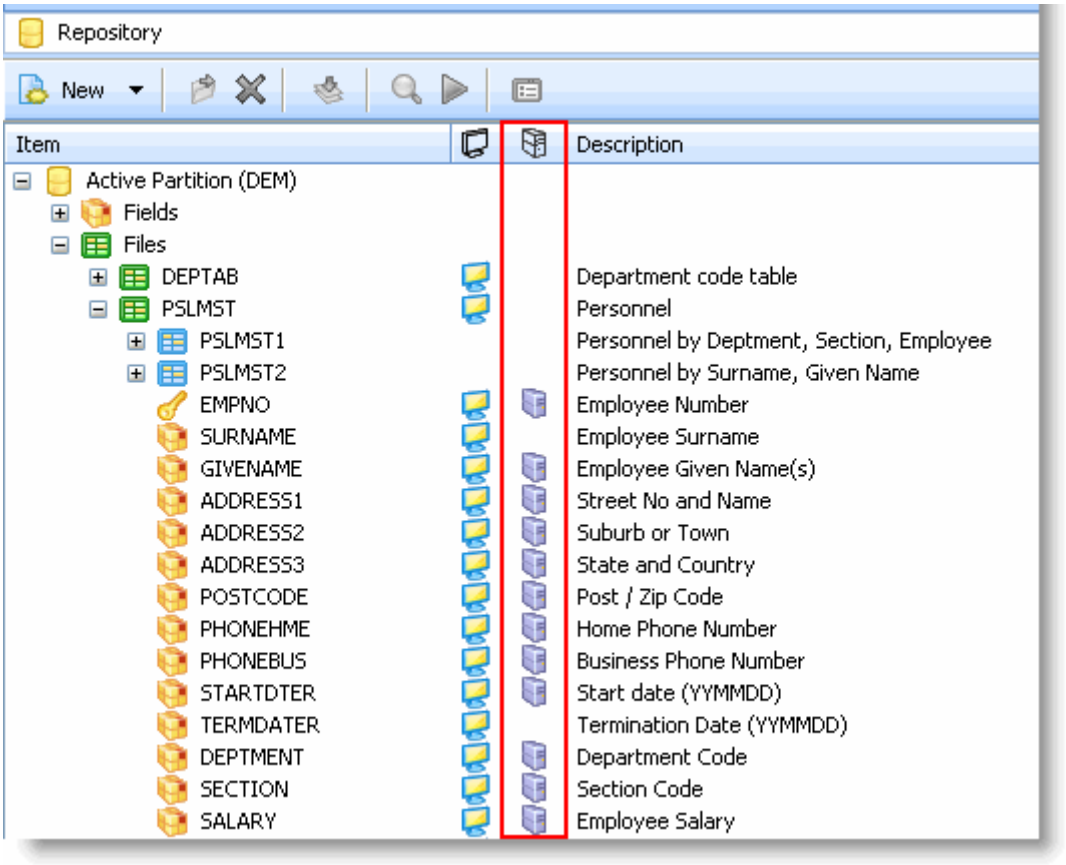
Use DBSS= parameter to change the maximum number of statements to cache. DBSS=0 will give the same behavior as previously.

Reusable SQL statements previously only included those generated for INSERT, and the SQL generated for re-read of the last record for DELETE and UPDATE commands. This enhancement includes generating reusable DELETE and UPDATE SQL for most files.

➢ Faster and More Detailed Information about Objects in Master Repository

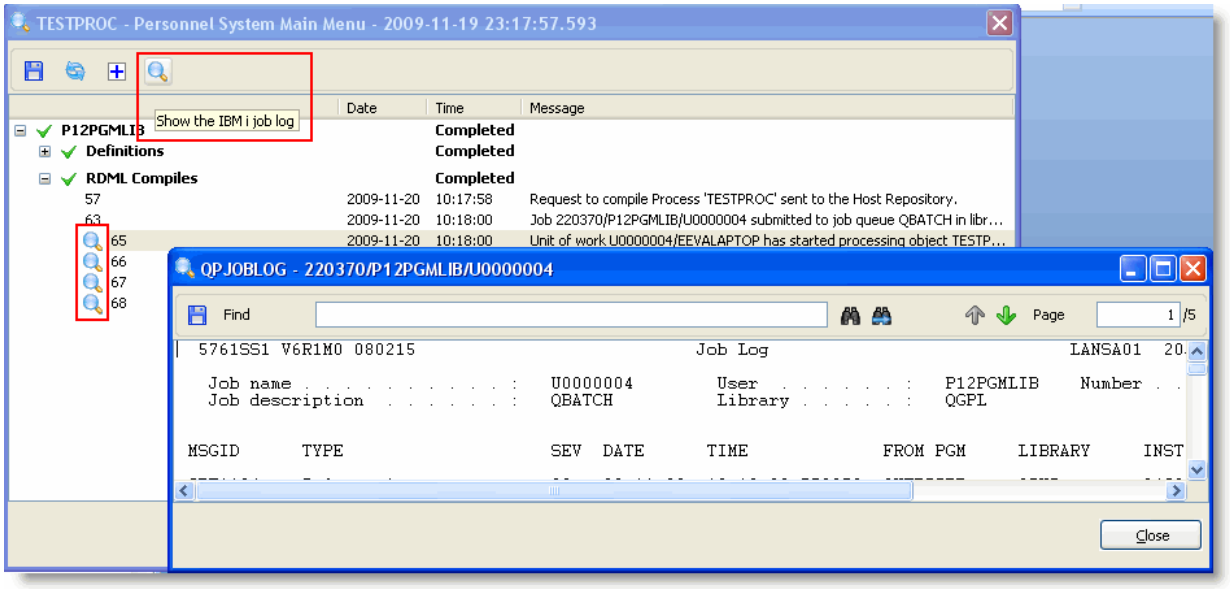
The Repository tab in a slave system's Visual LANSA IDE now provides detailed information about objects in its master repository, such as fields in a file, file versions, field length, type, RDMLX flag, task locked to, etc.

Refreshing the Master repository data is now considerably faster than in the previous version. The data is stored and reloaded each time Visual LANSA is started, and when you refresh the contents of the Repository tab, only information about objects that have changed in the master is retrieved. For more information, see Refresh Master Object List in the Administrator Guide.



➢ Check-in Joblog Viewer

Job logs created on the master for check-in jobs can be viewed in the Check-in Joblog Viewer. The whole file is downloaded to Visual LANSA and displayed one page at a time.



You can save the job logs and search them. Toolbar buttons are provided for paging up or down, or you can enter the specific page number.

<u>You can also navigate around the spool file using the keyboard:</u>

- Page Up – Show the previous page
- Page Down – Show the next page
- Ctrl+Home – Show the first page
- Ctrl+End – Show the last page

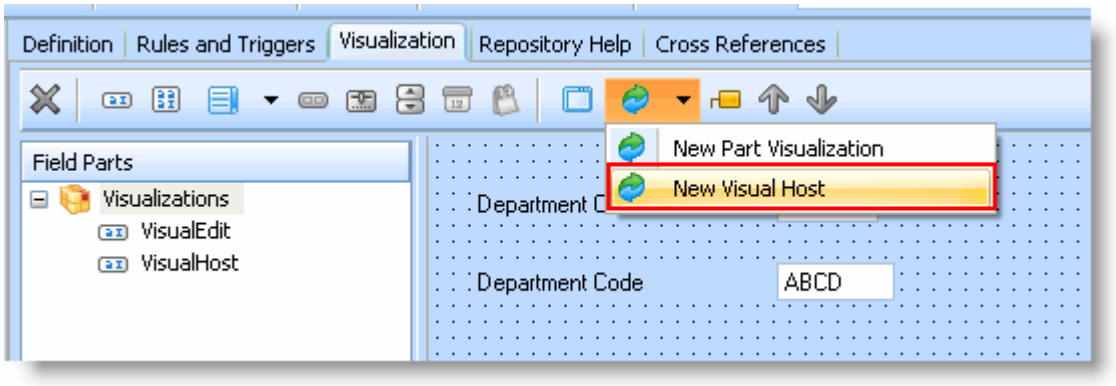
Arrow up and down move the cursor up and down. They will also scroll the spool file one line at a time when at the top or bottom of the display area. This means that it is possible to see the bottom of one page and the top of the next.

➢ Field Visualization Enhancements

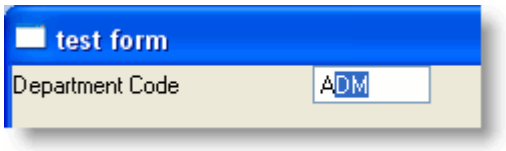
1. *Reusable Parts as Edit Portion of Field*

You can define a reusable part to act as the edit portion of a field. In this way you have all the flexibility of a reusable part while still taking advantage of LANSA Repository features for multilingual descriptions, labels and column headings.

In this example a reusable part has been created to handle auto complete for a field Department. The part is then associated with the field visualization a *visual host:*

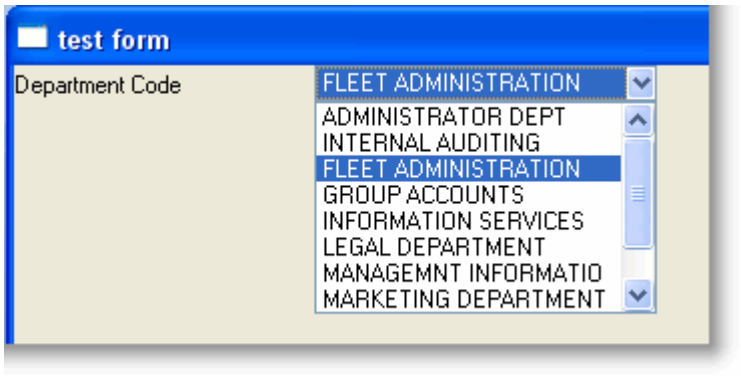


When the field is used on a form, the label is determined by the field definition, but the edit portion of the field uses the autofill logic created in the reusable part.



2. *Dynamic Pick Lists*

You can define dynamic pick lists which use the logic in a reusable part to, for example, load the values in the list from a file at runtime.



➢ New Intrinsics in RDMLX

1. Floating Point Intrinsics

The following floating point mathematical intrinsics are now available:

Trigonometric:
- sine, arcsine, sineh,
- Cosine, arccosine, cosineh
- Tangent, arctangent, tangenth, arctangent2

Logs, etc:
- Exponential, logarithm, logarithm10
- Power, sqrt

Others:
- Fabs, fmod,
- IsNANorND

Certain intrinsics may return values that are not recognized numbers, such as arccosine(3). The value returned could be either a **nan** (not a number) or an **nd** (non-determinate). -1.#IND
Isnanornd() can be used to check if the result is not a number.

Note:
The trigonometric intrinsics assume the input value is in radians, not degrees.

2. Binary String Conversions

A hexadecimal string cannot be represented in a numeric type, so the Binary String primitive is used as a staging area for the conversion.

Integers can be converted to and from hexadecimal strings by using the following intrinsics:
- Given an integer, convert to binary string using #integerValue.AsBinaryString(),
- Convert the binary string to a hexadecimal representation using #binayString.AsHexString(),
- Given a binary string containing a hexadecimal value, access as an integer value using #binaryString.AsHexToInt().

These methods work with both integers and long integers.

Example
Define Field(#myRBStr) Type(*BIN) Length(128)
Define Field(#L8Int) Type(*INT) Length(8)
#myRBStr := (9999).AsBinString()
#myRBStr.AsHexString() gives 0F270000
#L8Int := 169999999999
L8Int.AsBinString().AsHexString() gives FF23CA9427000000 on intel-based computers

3. *Integer Intrinsics*

Two new intrinsics are available for integer/longlong, Mod and Div.

➢ Set Value of Unknown Fields in Lists and Grids

You can set the value of a field without knowing its name in columnar views such as ListBox, GridView, ComboBox, Graph and unlevelled Tree. You use the SetValueAt() method and a ValueAt property to assign and retrieve the display string by referencing the rows and columns.

PRIM_EVP and PRIM_EVEF primitives can have values assigned using the SetValue() method.

If the value supplied is not allowed for the data type of the targeted element (for example, a string "abc" for a number), the SetValueAt returns LP_FALSE. If the assignment is successful, LP_TRUE is returned.

In the following examples, the various #OutputValueN are instances of #STD_TEXTL.Visual.

SetValue Examples using Primitives

```
Define_Com Class(#PRIM_EVP) Name(#evpTmp) Reference(*DYNAMIC)
If (#STD_NUM.SetValue( "a.bc" ) <> True)
#OutputValue1 := "not a number"
Endif
#evpTmp <=  #OutputValue2
#evpTmp.SetValue( "def" )


#OutputValue3.Assign( "my value" )
(#OutputValue4 *As #PRIM_EVEF).Assign( "hello" )


Define_Com Class(#PRIM_VAR) Name(#myVariant)
#myVariant.SetValue( "123.45" )


Define_Com Class(#prim_boln) Name(#myBoolean)
#myBoolean.SetValue( "true" )
```

SetValueAt Examples using Columnar Views

```
#TreeV1.SetValueAt( 1 1 "tv1" )
#TreeV1.Items<2>. SetValueAt ( 1 "tv1" )
#TreeV1.columns<1>. SetValueAt ( 3 "tv1" )

#Grid1. SetValueAt ( 1 1 "11" )
#Grid1.Items<2>. SetValueAt ( 1 "21" )
#Grid1.columns<1>. SetValueAt ( 3 "31" )

If (#graph1. SetValueAt ( 2 1 "abc" ) <> True)
#OutputValue1 := "invalid value for graph"
Endif

#Grid1.Cell<1 1>. SetValueAt ( "8" )
#Grid1.Cell<1 2>. SetValueAt ( "9" )

#Grid1.Cell<2 1>. SetValueAt ( "1" )
#Grid1.Cell<2 2>. SetValueAt ( "2" )
```

Restrictions

Only lists/views supporting columnar views provide this functionality. Leveled trees cannot be used: a runtime exception is returned.

Memo lists do not provide this functionality.

Lists must have been previously initialized with row data before assigning new values. For example, Inz_List Named(#TreeView) Num_Entrys(2).

Using indices that are out of bounds result in a runtime error indicating that the index is invalid.

Using ValueAt

Columnar Views provide access to their members using the ValueAt property. This takes the row and column index for the cell as parameters.

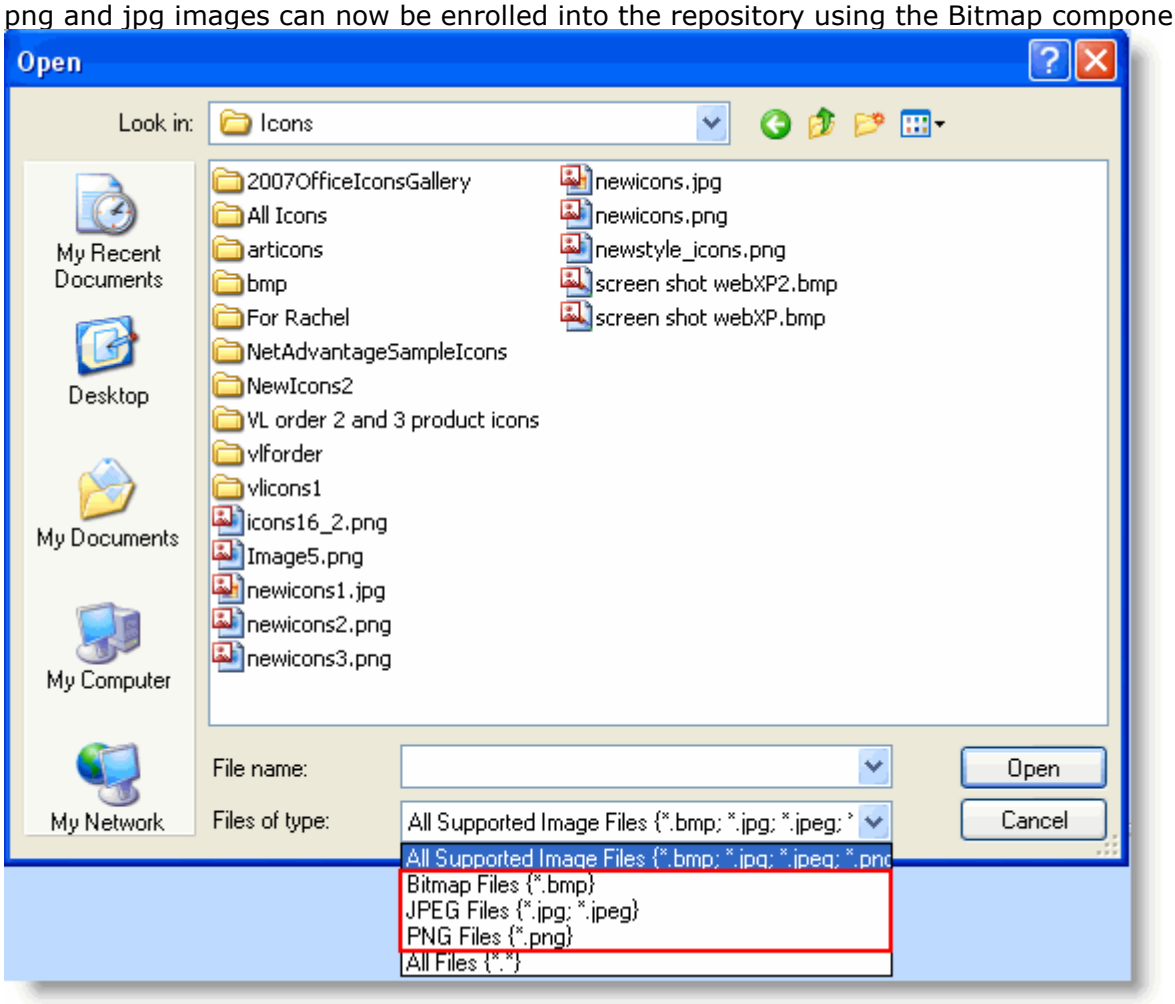
Example

This is an example of how SetValueAt() can be used:

```
Begin_Loop Using(#myRowIdx) To(3)
  Begin_Loop Using(#myColIdx) To(#LBox1.columns.itemcount)
    #myStringVal  :=  #LBox1.ValueAt<#myRowIdx #myColIdx>
    * do something useful with #myStringVal …
  End_Loop
End_Loop
```

➢ Enhanced Image Support

png and jpg images can now be enrolled into the repository using the Bitmap component:

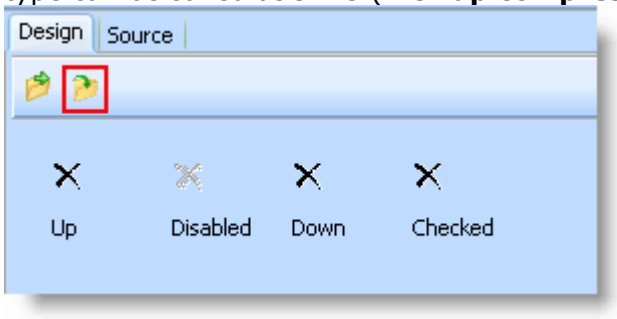


Also there is a **new Bitmap Editor** to view and manage image files. The ability to copy an image component to a new one was added, so now you may export BMP, GIF and JPGs to a hard disk file and subsequently load as a new image resource. You can also view basic image properties such as type and compression.

Further, following image and file types are supported:

- BMP to RGB, BITFIELDS, RLE8 and RLE4 compressions
- JPEG to JFIF and EXIF file types

A bitmap component can now be saved to a file, and BMP images of RGB and BITFIELDS type can be saved as JPEG (**Bitmap compression support**):



➢ Multi-Monitor Support

New properties have been added to Visual LANSA applications to support multiple monitors.

To cater for the unplugging of a secondary monitor or a corrupt monitor registry setting, you can set the PRIM_FORM.EnsureVisible property to Yes in your forms. When this property is used and a monitor is no longer available when registry values for the form are loaded, the form is moved back into the virtual screen. Note this property has to be set before the form is realized.

An easy means of turning on the EnsureVisible functionality for your whole application is to set the SYS_APPLN.EnsureVisibleForms property to True. In this case you would set the EnsureVisible property of the forms to the default value Application.

Design Time Considerations
To support multiple monitors, a "logical primary monitor" concept has been introduced. The monitor in which the Editor is shown is the logical primary monitor. If the Editor is on a secondary monitor (with a left sample left of -1280), any designed forms will be offset by this amount to ensure the Left and Top properties do not contain incorrect negative values.

So if your Editor is on a secondary monitor, all your forms will also be designed there and their bounds coordinates will be offset so that the runtime values will be correct for the primary monitor.

If you have been operating the Editor from a secondary monitor in an earlier Visual LANSA version, you will need to remove any saved incorrect co-ordinates from the source code for the forms to display properly.

Other Monitor Properties
- *SYS_APPLN.Monitors* provides a collection of all the monitors attached to the desktop.
- *SYS_APPLN.PrimaryMonitor* provides the primary monitor attached to the desktop.
- *SYS_APPLN.VirtualScreen* properties provide the dimensions of the entire desktop area.

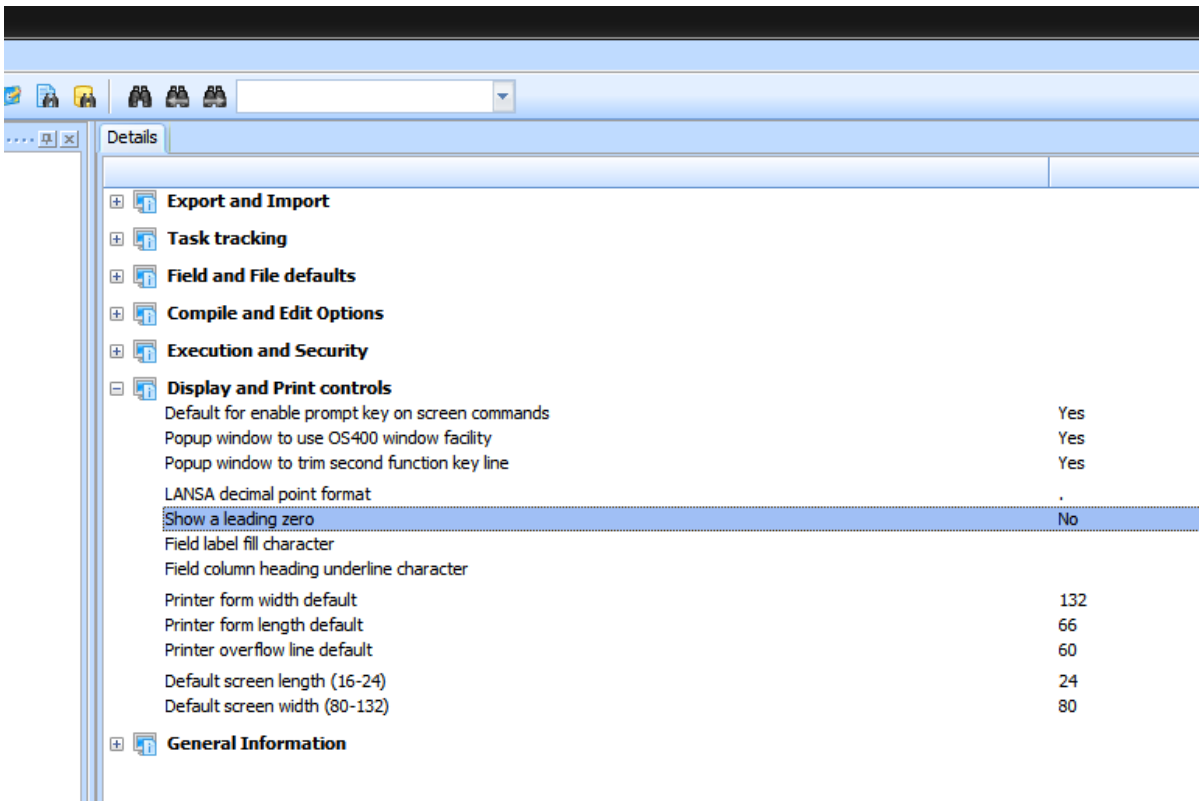
You can use these properties if you have a preconfigured and hardwired monitor layout and want to implement your own behavior to ensure forms position correctly.

➢ Regional Benefits

Keyboard:
Allow numeric keypad '.' decimal when regional settings have ',' as decimal.

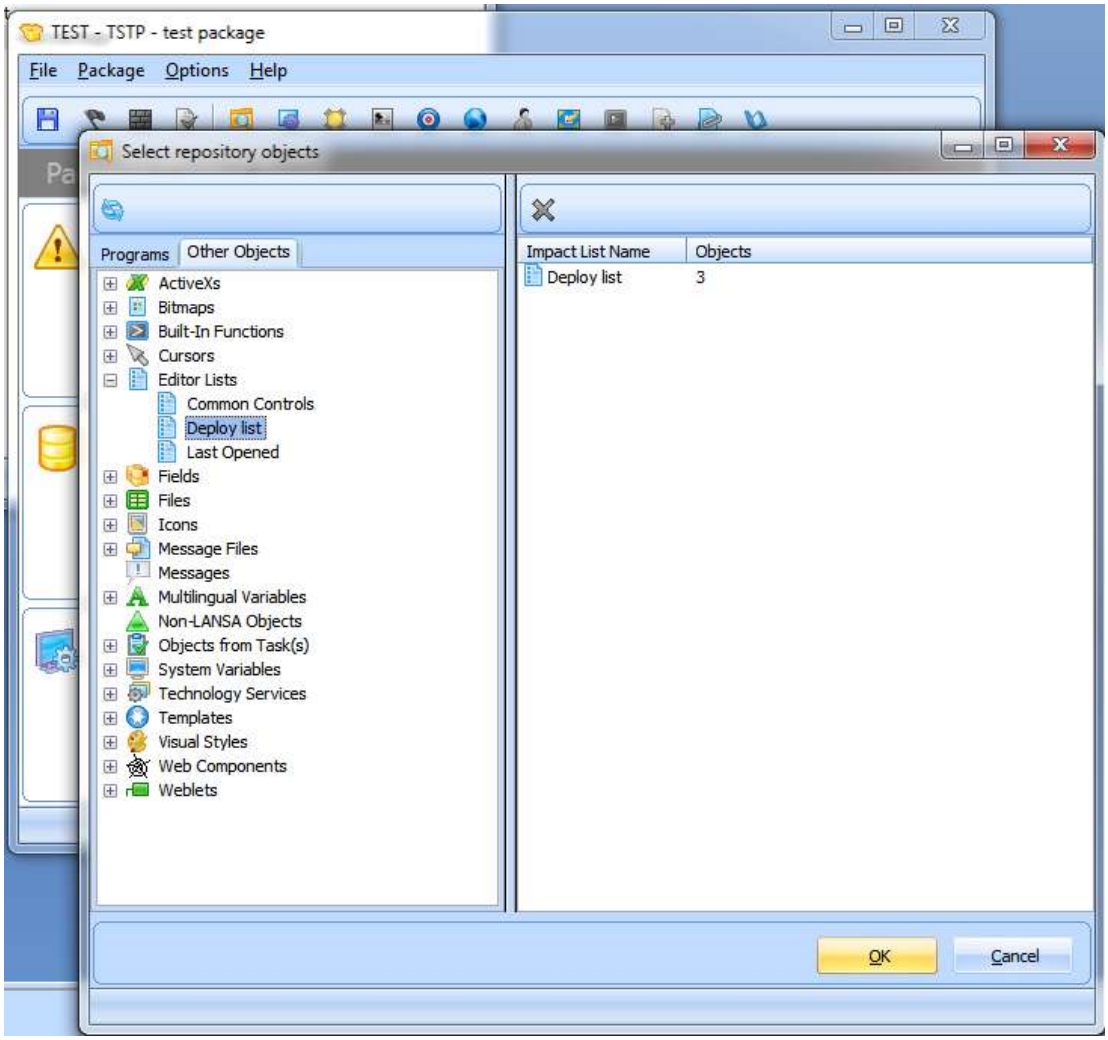
Fields:
Edit codes in Visual LANSA to show a leading zero, e.g. 0.25 and 0,25.

Setting "Show a leading zero" to "Y" indicates that Visual LANSA code is to display numeric values containing decimals with a leading zero. For example 0.12 or 0,12 depending on the LANSA decimal point format. This setting would be suitable when IBM i system value QDECFMT is "J". IBM i supports this via the DC@OSVEROP data area setting of *LEADZERO=1 which can be set via the Display and Print Controls "Show a leading zero" value accessed from System Settings from the Administration Tasks menu.



➢ Enhanced Deployment

New selection of objects from Editor Lists
The Deployment Tool object selection now supports **Editor Lists**. This introduces much greater flexibility into the selection of objects as Editor Lists can use wild card selection as well as supporting filters for modified date, task id, user and various other criteria. Editor List objects are added to a Deployment Package when the Editor List is selected (not at build time).



New templates to Web enable application environment
Three new deployment templates have been supplied to support the delivery and upgrade of a web enabled application. These are:

- XWEBENB
- XWBEVENB
- XOTHOBJ

Refer to documentation in the LANSA Application Deployment Tool guide for more information.

Support for LANSA Version 10

With the release of Version 12, LANSA will be discontinuing maintenance (i.e. software changes) of Version 10 on May 31, 2010 (maintenance for all versions prior to Version 10 was discontinued with the release of Version 11).

Though maintenance for LANSA Version 10 will be discontinued on May 31, the LANSA Support team will continue to support and help you work through any questions or issues you might be facing with V10 or earlier releases. However, if the issue requires modification to the underlying LANSA source code or if it relates to a bug that was fixed in a later release, an upgrade would be required.