LANSA NEWSLETTER

LANSA V16

We are very proud to announce LANSA Version 16. It contains lots of new features and enhancements that are detailed it in this Newsletter.

But there is more good news. We just have also released a new version of Composer and LANSA BI.

INSIDE THIS ISSUE

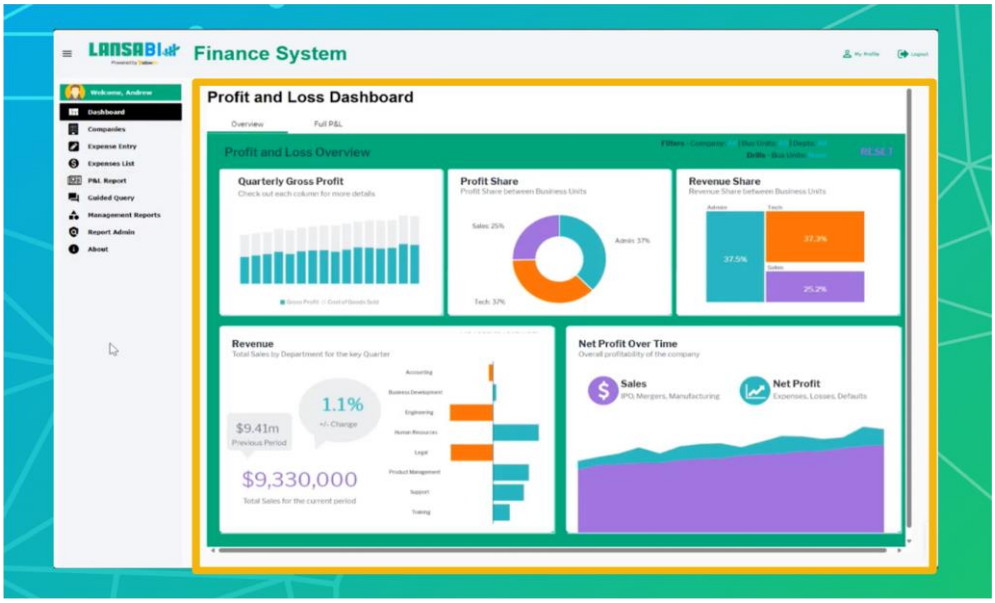
LANSA V16

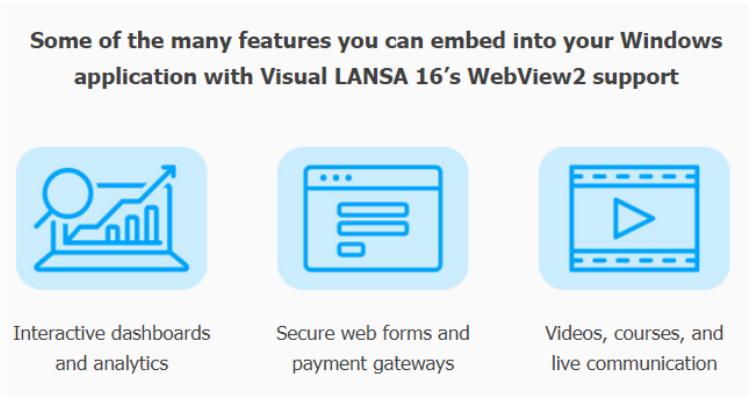
What's new in Visual LANSA V16?

1.

REST API and third party libraries integration via WebView2

Visual LANSA Version 16 integrates WebView2, a Chromium-based browser, to support the latest HTML5, CSS, and other modern web standards, expanding functionality within the IDE using 3rd party libraries and REST APIs integration and future-ready the application.



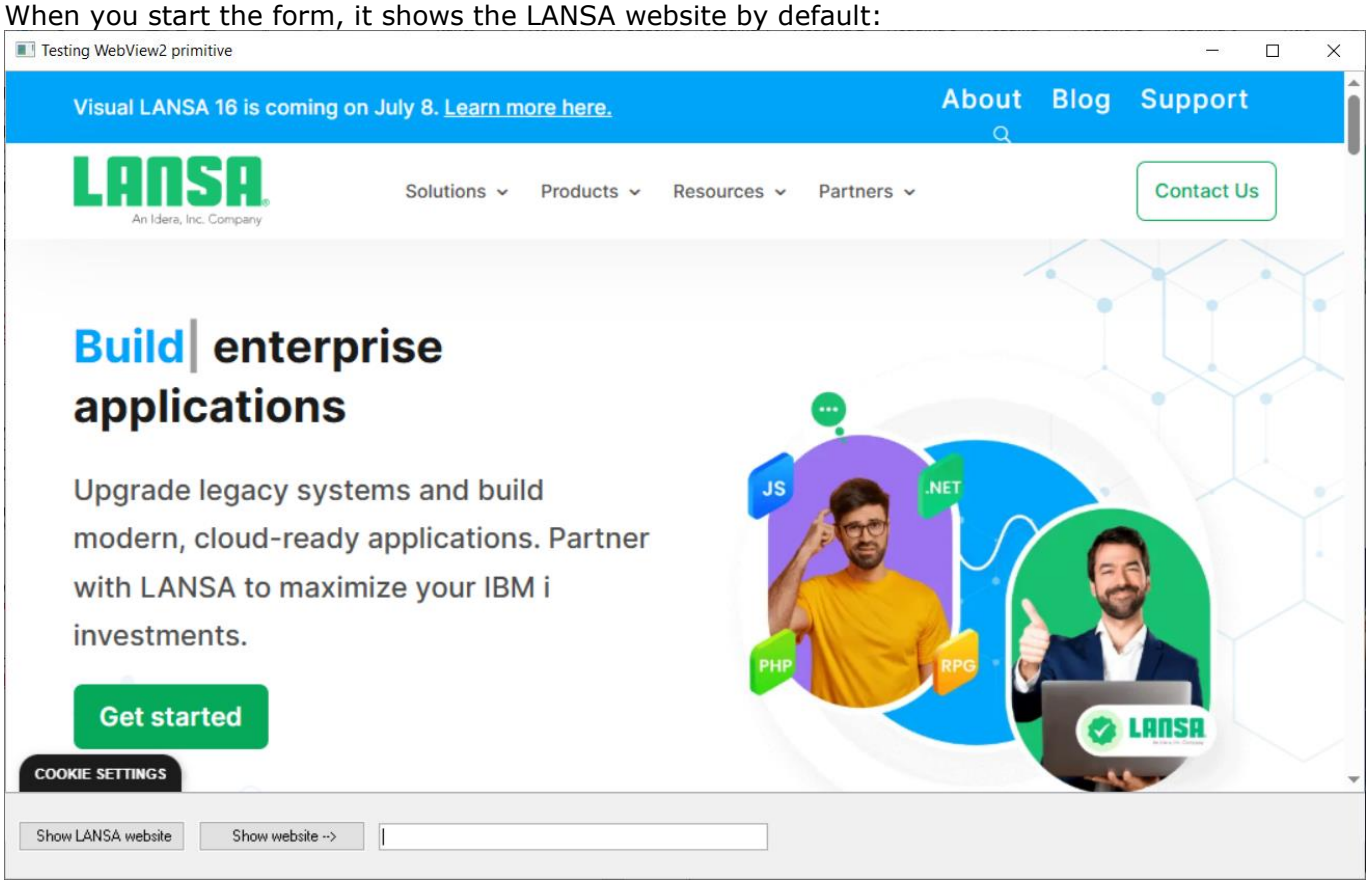


The next page shows an example how to use this new primitive.

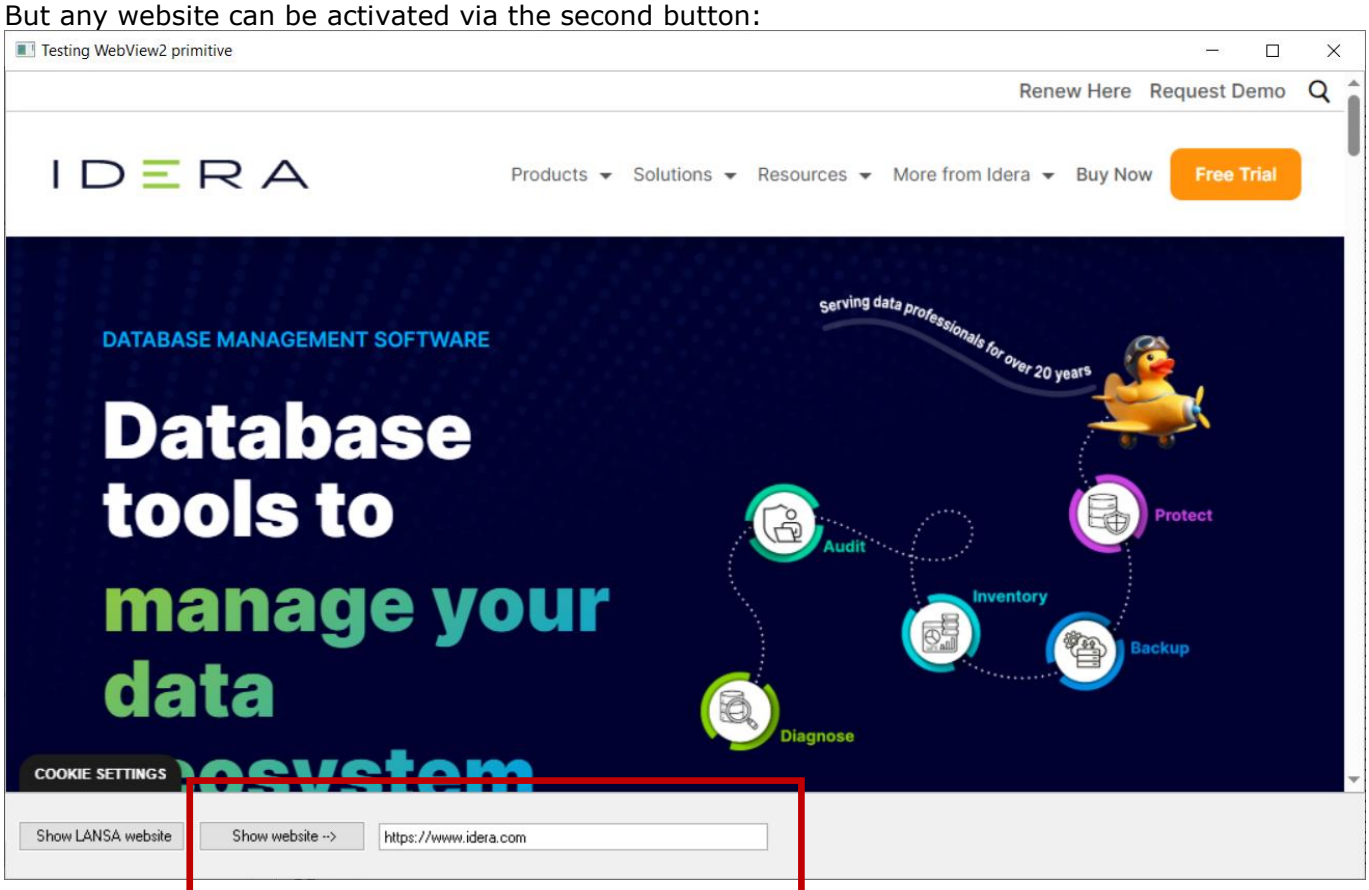
You can copy/paste source example below (you have to create a new Form component):

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientwidth(1048) Clientheight(586)
Componentversion(2) Left(273) Top(220) Layoutmanager(#LayoutMain)

Define_Com Class(#PRIM_TBLO) Name(#LayoutMain)
Define_Com Class(#PRIM_TBLO.Column) Name(#LayoutMainColumn1) Displayposition(1)
Parent(#LayoutMain)
Define_Com Class(#PRIM_TBLO.Row) Name(#LayoutMainRow1) Displayposition(1)
Parent(#LayoutMain) Height(1.65)
Define_Com Class(#PRIM_TBLO.Row) Name(#LayoutMainRow2) Displayposition(2)
Parent(#LayoutMain) Height(66) Units(Pixels) Minheight(66) Maxheight(66)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem1) Manage(#PanelWebv)
Parent(#LayoutMain) Row(#LayoutMainRow1) Column(#LayoutMainColumn1)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem2)
Manage(#ShowLANSAWebsite) Parent(#LayoutMain) Row(#LayoutMainRow2) Sizing(None)
Column(#LayoutMainColumn1) Alignment(CenterLeft) Flow(Right) Marginleft(10)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem3)
Manage(#ShowOtherWebsite) Parent(#LayoutMain) Row(#LayoutMainRow2) Sizing(None)
Column(#LayoutMainColumn1) Alignment(CenterLeft) Flow(Right) Marginleft(10)
Define_Com Class(#PRIM_TBLO.Item) Name(#LayoutMainItem4) Manage(#STD_TEXTS)
Parent(#LayoutMain) Row(#LayoutMainRow2) Sizing(None) Column(#LayoutMainColumn1)
Alignment(CenterLeft) Flow(Right) Marginleft(10)

Define_Com Class(#PRIM_WEBV) Name(#PanelWebv) Displayposition(1)
Parent(#COM_OWNER) Tabposition(1) Top(0) Height(520) Width(1048) Left(0)
Defaulturl('https://www.lansa.com')
Define_Com Class(#PRIM_PHBN) Name(#ShowLANSAWebsite) Caption('Show LANSA
website') Displayposition(2) Parent(#COM_OWNER) Tabposition(2) Top(541) Width(129)
Height(24)
Define_Com Class(#PRIM_PHBN) Name(#ShowOtherWebsite) Caption('Show website -->')
Displayposition(3) Left(149) Parent(#COM_OWNER) Tabposition(3) Top(541) Height(24)
Width(129)
Define_Com Class(#STD_TEXTS.Visual) Name(#STD_TEXTS) Componentversion(1)
Displayposition(4) Height(21) Parent(#COM_OWNER) Tabposition(4) Top(543)
Usepicklist(False) Width(300) Labelposition(None) Marginleft(0) Left(288)

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

Evtroutine Handling(#ShowLANSAWebsite.Click)
#PanelWebv.NavigateTo Url('https://www.lansa.com')
Endroutine
Evtroutine Handling(#ShowOtherWebsite.Click #STD_TEXTS.Enter)
#PanelWebv.NavigateTo Url(#std_texts)
Endroutine
End_Com
```

When you start the form, it shows the LANSA website by default:



But any website can be activated via the second button:



Using Webview2

Component Definition

The Visual LANSA Primitive for WebView2 provides standard web page controls (Navigate, Back, Forward, Stop, and Reload) along with access to navigation events, such as starting and completing, to help manage and customize your UI.

```
Define_Com Class(#PRIM_WEBV) Name(#WebView) Parent(#COM_OWNER)
Displayposition(1) Tabposition(1) Height(890) Top(116) Left(0) Width(1659)
Defaulturl('https://docs.lansa.com/16/en/')
```

Properties

CanNavigateBack
Returns a Boolean as to whether the browser can navigate back in history.

CanNavigateForward
Returns a Boolean as to whether the browser can navigate forward in history.

DocumentTitle
Returns a string that is the title of the current HTML document.

DefaultURL
Sets the URL which will load once the WebView is initialized.

Source
Returns the current URL in the WebView component.

Here is an example of using these properties to control your UI:

```
Mthroutine Name(ToggleControls)

    #BTN_Back.Enabled := #WebView.CanNavigateBack
    #BTN_Forward.Enabled := #WebView.CanNavigateForward
    #URL := #WebView.Source
    #COM_OWNER.Caption := #WebView.DocumentTitle

Endroutine
```

Methods

NavigateBack
Navigates back in history.

```
Evtroutine Handling(#BTN_Back.Click)
    #WebView.NavigateBack
Endroutine
```

NavigateForward
Navigates forward in history.

```
Evtroutine Handling(#BTN_Forward.Click)
    #WebView.NavigateForward
Endroutine
```

NavigateTo
Navigates to the URL input as the URL parameter.

```
Evtroutine Handling(#BTN_Go.Click)
    #WebView.NavigateTo( #COM_SELF.FormatURL )
Endroutine
```

Reload
Reload the current webpage.

```
Evtroutine Handling(#BTN_Reload.Click)
    #WebView.Reload
Endroutine
```

Stop
Stops the loading of a webpage.

```
Evtroutine Handling(#BTN_Stop.Click)
    #WebView.Stop
Endroutine
```

Events

NavigationComplete
Fired when the loading of a webpage has finished.

```
Evtroutine Handling(#WebView.NavigationCompleted) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
    #Navigating := False
    #COM_SELF.ToggleControls
    #URL := #WebView.Source
Endroutine
```

NavigationStarting
Fired when the Loading of a webpage commences.

```
Evtroutine Handling(#WebView.NavigationStarting) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
    #Navigating := True
    #COM_SELF.ToggleControls
Endroutine
```

2.
Cloud-ready Azure licensing

Enhanced Azure Cloud licensing capabilities now include support for cloud-based licenses, streamlining deployments and aligning with modern cloud-first architectures.

Cloud Account ID Licensing

You may create a Visual LANSA runtime key that is tied to an Azure or AWS account ID instead of a specific Virtual Machine (VM). Any VM running in that Cloud account with the license file installed will pass the license check.

Requirements

- Visual LANSA V16
- Visual LANSA V15 with EPC150070 and Patch EPC150070HF_24111

Both must be installed to generate a license that is tied to the Cloud account ID. As part of this implementation, the license file will need to be deployed in a Cloud image or Virtual Machine. You will have multiple options for deployment.

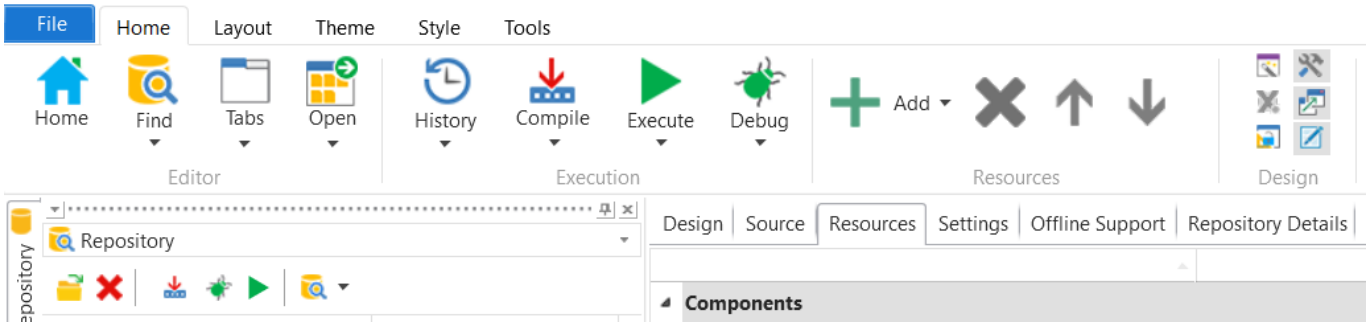
If you have two or more Cloud accounts for the same Cloud provider, all files can either be installed in the one Machine Image so that it may be used in either account, or the file(s) could be deployed with the application. Any Cloud instance running in the account will be licensed, as the license is locked to the account ID. The same license file will be used for all Cloud VM instances in an account.

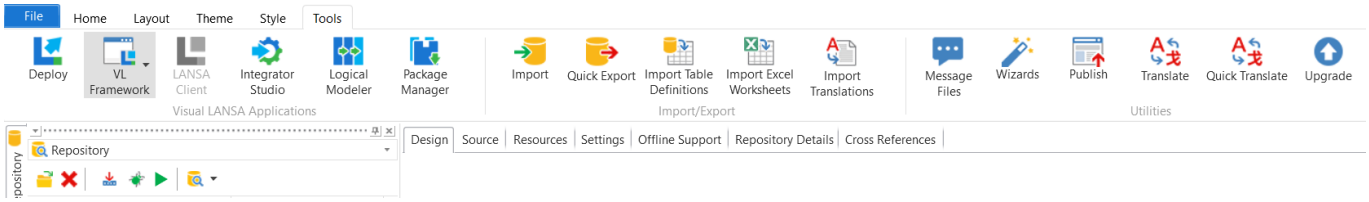
The number of licensed instances will be defined in the licensing contract.

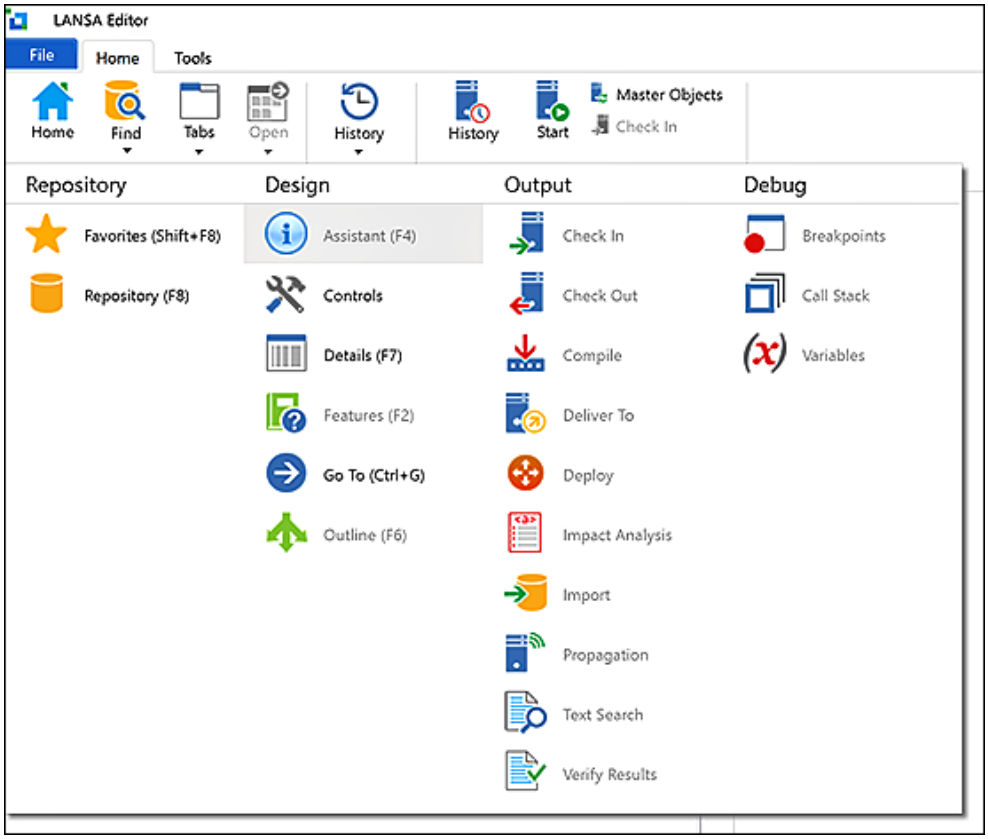
3.

Refreshed toolbar iconography

V16 includes the first phase of a visual update that modernises application interfaces with refreshed iconography for a more contemporary and user-friendly design.



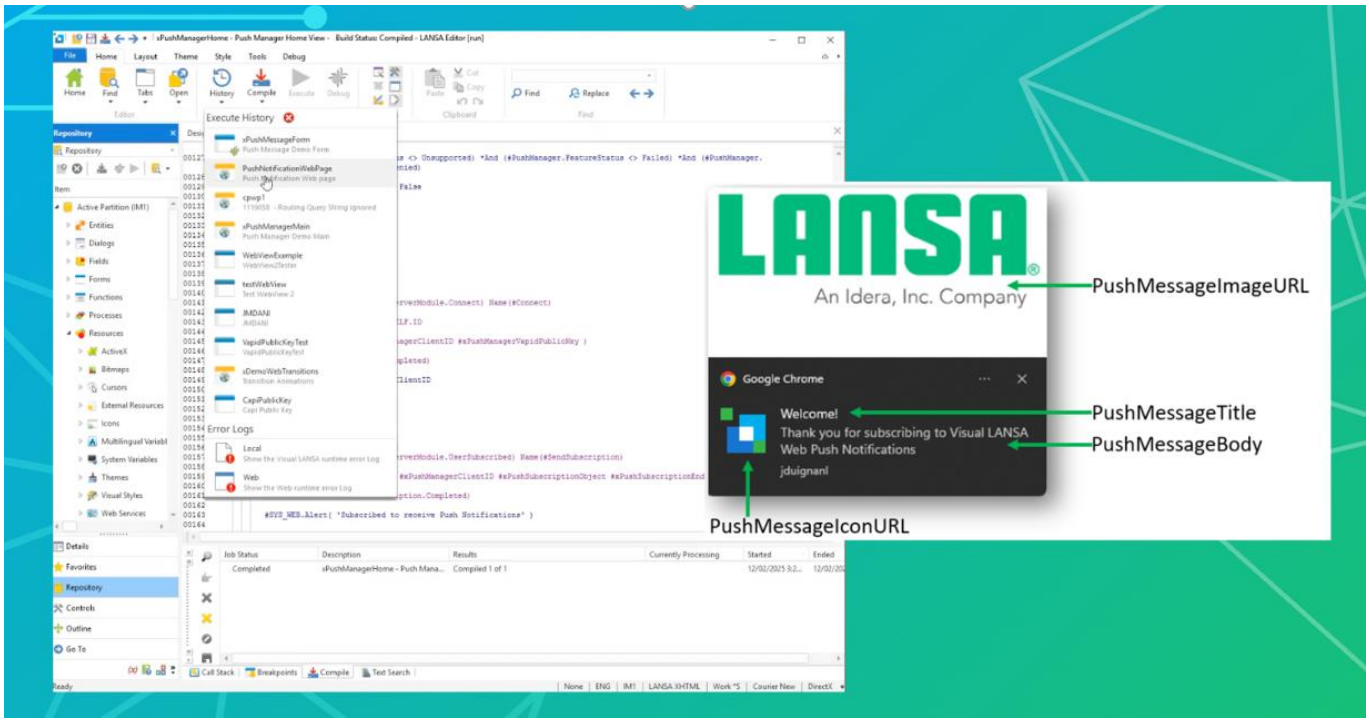




4.

Real-time push notifications from server to client

Simple and easy method for developing real-time Push Notifications triggered from server to client allowing encrypted data communication between server and client.



Visual LANSA is integrated with push notifications to keep you informed about important updates from apps.

The push notifications lifecycle is the following:

- Request Permission to send push notifications.
- A user subscribes to receive push notifications from a web application resulting in a subscription object.
- The web application passes the subscription object to the application server to store it in a database.

- An event is triggered on the application server to send a notification. The subscription objects are fetched from the database.
- A push notification request is sent to the user agent.
- The user agent processes the request.
- The end user receives the notification.

LANSA simplifies this entire process by making it easy to get updates:

- The Visual LANSA Web Push Manager allows users to easily subscribe to receive notifications and results in the subscription object.
- The subscription object is passed back to the server module and stored in a database table.
- LANSA defines an event for sending a notification. For example, the system will be undergoing maintenance at a given time and date, there is a sale on a product or a set of tasks have been completed.
- The subscription object can then be fetched from the database table on demand to construct a custom notification for the end user.
- The notification is sent from your LANSA application.

The Visual LANSA Push Notification feature simplifies the enhancement of your Progressive Web Applications (PWA). It removes the complexities associated with integrating multiple systems, languages, and technologies, along with the need for custom encryption and managing data across different platforms.

To start using this feature you need to enable the Visual LANSA PWA web application and an EC256 private key.

Key Components

Web

PRIM_WEB.PushManager
Controls a client push subscription providing a method for a user to give permissions and subscribe to receive push notifications. The resulting subscription object is sent to the server and stored so it can be used once.
For detailed information, go to Push Manager.

PRIM_WEB.PushManager.Subscription
Contains the necessary information that is returned to the Application Server to store the details of a user's subscription.
For detailed information, go to Push Manager Subscription.

Application Server Windows/IBM i

PRIM_SRVM.HttpPushMessage
Sends push notifications to subscribed users based on their subscription details.
For detailed information, go to Push Message.

PRIM_CAP.PrivateKey.VapidPublicKey
Retrieves the Vapid public key, which is used as the application server key.
For detailed information, go to Vapid Public Key.

5.
OAuth 2.0 Email Authentication

Secure authentication via OAuth 2.0 for SMTP and POP3 ensuring better protection against data breaches and providing easier to manage access.

For more information refer to IMAPMailService, POP3MailService, SMTPMailService, and SMTPMailAttachmentSignatureService.

6.
New IBM QPWDRULES password rules

Starting with LANSA V16 for IBM i, the installation process has been updated to support IBM system password settings (QPWDLVL and QPWDRULES).

Historically, the user profile created as part of the LANSA on IBM install had a simple password of lansa (originally) and vlansa (V14 and V15).
More recently, we have encountered situations with new customers and prospects that have stricter rules regarding passwords for their user profiles (minimum length, allowed number of numerics or special characters etc.)

V16 Install change

We have put a change into the V16 IBM install to cater for the IBM system values that control user profile password levels and rules.

That is:
QPWDLVL https://www.ibm.com/docs/en/i/7.3.0?topic=passwords-password-level-qpwdlvl
and
QPWDRULES https://www.ibm.com/docs/en/i/7.3.0?topic=passwords-password-rules-qpwdrules

We have elected to implement a program that will check the rules and create the user profile with a password that will adhere to the local policies.

7.
New HttpClientRequest

In LANSA version 16 there is a new http client that can be used to do http calls: **#PRIM_SRVM.HttpClientRequest**.

Big advantage of the new client is that it is in the PRIM library so it can use the PRIM objects related to JSON handling. The use of PRIM objects to write and read JSON generally results in a better performance compared to the XPRIM libraries that can also be used to read and write JSON.

The new client is defined like this:

```
Define_Com Class(#PRIM_SRVM.HttpClientRequest) Name(#HttpRequest)
```

For the rest, the usage of the client is the same as the #XPRIM_HttpRequest that was already available.

```
00070       Define_Com Class(#XPRIM_UriBuilder) Name(#Uri)
00071       Define_Com Class(#PRIM_SRVM.HttpClientRequest) Name(#HttpRequest)
00072       Define_Com Class(#PRIM_JSON.Document) Name(#JSONResponse) Reference(*DYNAMIC)
00073
00074       #Uri.SetScheme Scheme("http")
00075       #Uri.SetHost Host("localhost")
00076       #Uri.SetPort Port(8083)
00077       #Uri.SetPath Path("dem/icswsjwt/orders")
00078
00079       #HttpRequest.DoGet URL(#Uri.AsString.AsNativeString)
00080
00081     If (#HttpRequest.Response.IsSuccessfulRequest)
00082       If (#HttpRequest.Response.IsSuccessHttpStatusCode)
00083
00084         #JSONResponse <= #HttpRequest.Response.AsJson
00085         If (#JSONResponse.IsArray)
00086           For Each(#iOrder) In(#JSONResponse.AsArray)
```

In the sample on previous page you can see a #XPRIM_UriBuilder (lines 74-77) has been used to create the URL that is used by the HttpRequest on line 79. The checks if request was executed successfully (lines 81-82) are followed by putting the response JSON into a #PRIM_JSON.Document (line 84).

Previously PRIM_JSON objects needed to be added to the #XPRIM_HttpRequest as a string which something was a problem because a string is limited to 65535 characters. As shown in the sample the new http client can set and get PRIM_JSON objects directly so that limit is no longer there.

The #PRIM_SRVM.HttpClientRequest also has a new method called appendString. This allows you to add multiple strings to a http request. With that it is also possible to avoid the limit of 65535 characters in a string.

8.
Other new Features and Changes

Over 80 extensive updates across the IDE, runtime, and web environments, significantly boosting stability, compatibility, and developer experience/environments.

Highlights include:

- **Updated Support for Key Libraries and Standards**
  Enhanced support for MySQL 8.0 and 8.4 LTS, including improved handling of NChar, NVarchar, and SUNI columns, ensuring compatibility with modern MySQL environments and eliminating key integration failures.
- **OpenSSL Upgrade for Security Compliance**
  Upgraded to OpenSSL 1.1.1g, delivering improved encryption, secure communications, and alignment with current security standards.
- **Improved Session and Memory Management**
  Reworked a couple of internal processes to address memory leaks, REST API inefficiencies, and unstable session behaviours—particularly under large loads or during long-running processes.
- **Web & API Stability**
  Improved debugging, tracing, and error handling for REST APIs and TP jobs. Fixes include session clean-up, improved exception tracking, and better runtime diagnostics.
- **UI/UX Enhancements**
  Resolved issues in lists, dropdowns, charts, calendars, and pickers, with improvements to scroll behaviour, tab order, keyboard navigation, and visual consistency across devices and screen modes.
- **Security & Standards Compliance**
  Upgraded to OpenSSL 1.1.1g for enhanced data protection and session security. Integrated support for MySQL 8.0 and 8.4 LTS, and reinforced SQL operations including BLOB and JSON handling.
- **Accessibility & Internationalization**
  Enhancements for screen readers, better keyboard focus behaviour, and improved support for multilingual content.
- **Platform Compatibility**
  Verified stability with IBM i 7.5 and refined support for multi-tier and tablet environments, including VLF-RAMP execution.
- **Performance Optimization**
  Reworked internal modules to reduce memory leaks, optimize resource usage, and improve responsiveness under load.

Job Number vs Web Job

While investigating a strange locking issue within a VL Web application, where different results were seen between Development and Test systems, we under covered a potential application design issue which could surface depending on the underlying configuration of a LANSA web server and the assumptions made by server side program logic.

It is important to stress that this only relates to Web applications. 5250 and Windows client/server applications are not affected by the web server configuration.

By default, a V15 or V14 SP2 LANSA web installation will be automatically configured so that the web serving jobs remain active, only eventually being ended after 500 uses;

Depending on the particular configuration there may be one or (more likely) many web serving jobs active at the same time. Each web serving job will be allocated a unique job number even on a Windows web server where the job number is emulated. For each web serving job and for each of those 500 interactions, the job number will remain unchanged. After 500 uses, or if a web serving job ends abnormally, the job number is not preserved - any new web serving job will have a new job number. However, at the other extreme it is possible to adjust the lifespan of the web jobs to be very short; it is in fact possible to set the system so that a web job is used only once and then immediately ends. Therefore in that scenario every single web job interaction will definitely have a new unique job number.

In the normal course of events, the state of the job number, and whether it is constant or always changing, is largely irrelevant; but if your program logic ***relies*** on the job number either remaining the same or being different on every interaction, the default configuration of the web server will affect how that application operates. And if Development, Test and Production have different configurations for the lifespan of web serving jobs then an application relying on the job number could behave differently in those environments. Remember that a web serving job with a long lifespan will respond to requests from many different clients / users - the connection is stateless.

One specific scenario where the job number is referenced is when a **LOCK_OBJECT BIF** is used. The LOCK_OBJECT BIF is used to place a "soft-lock" on an item of data. Although the BIF only requires the object type and a multi-part key to identify the item being locked, behind the scenes the information stored also includes the current job number. This is recorded to allow the BIF to check if a lock has already been placed by the same job (returning an IG status to tell the program that the lock is already in place for the current job).
If the web serving jobs only stay active for one use, then on each client / server interaction a new job number will be used. This means that if 2 different clients attempt to place a lock on the same object, because the job numbers are different the second client will get an ER response, i.e. the lock has been placed by another client.

However, if the web serving jobs stay active for multiple uses, then different client / server interactions could likely reuse the same job (especially in a system with light web traffic) and therefore the ***same*** job number will be used. In this scenario it means that if 2 different clients attempt to place a lock on the same object, because the job numbers are the same, the second client will get an IG response falsely indicating that the second client already had the lock, when they did not.

It is important to say that if your application does not reference the job number (either directly, for example via *JOBNUMBER system variable or indirectly via something like LOCK_OBJECT) then this potential issue will not effect you. It only becomes important if you are using the job number, and either expecting it to stay constant, or to be always different.

If there is an issue, resolving it depends on the specific nature of the application, the way the users interact with it, and whether you expect the job number to be constant or always different. There are various solutions which might be employed, for example here are some suggestions, but there may well be others which would be relevant to particular scenarios and applications:

1) If using LOCK_OBJECT and it makes sense in your application, treat an IG response as an ER (locked) then if the job number stays the same, different client interactions will always return a "locked" status on that item.
2) Generate a custom unique code or sequence number on the client and pass between client and server and use this in place of the job number (for example use it as part of the LOCK_OBJECT key).
3) If using LOCK_OBJECT / UNLOCK_OBJECT you could create your own soft-lock module (and the underlying table to keep track of locks) and replace the BIF calls with your own custom methods.
4) If you always require a different job number on each interaction, and the program logic cannot be changed, adjust the web server configuration to force web serving jobs to only be used once - HOWEVER beware as this will ***adversely*** affect the web serving job performance (jobs will be constantly ending and restarting) so the implications of this should be fully considered before making a change of this type, especially on a very busy system (and on the IBMi you might be creating a large volume of spool files).

Finally, regardless of the application and whether or not job number is used at all, it is always wise to define Development, Test and Production web serving environments in approximately the same way to ensure consistent results across all environments.

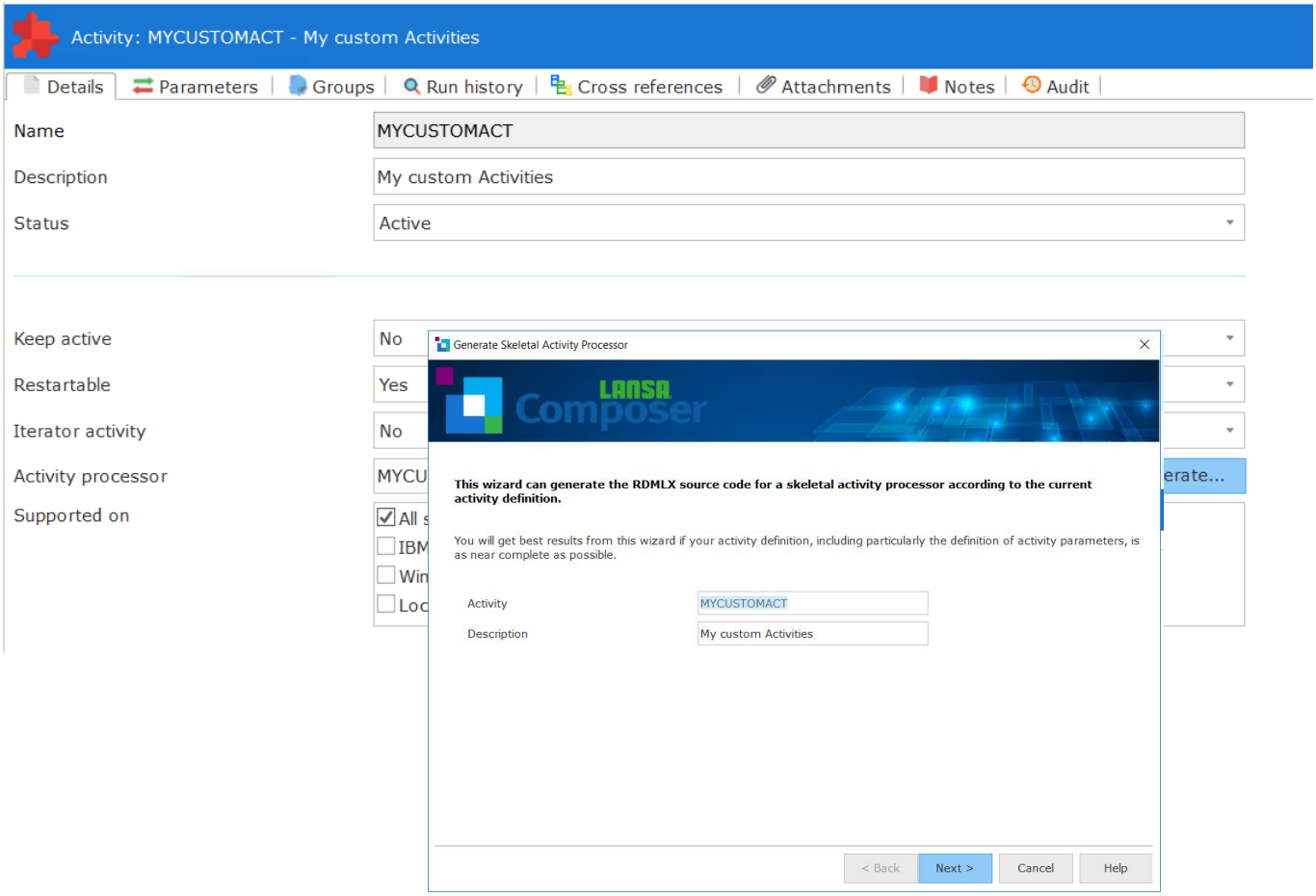
Composer 8

We're excited to announce the release of LANSA Composer 8.0, bringing enhanced compatibility and expanded support to empower more capabilities for your development and integration needs.

What's new in Composer 8?

Visual LANSA 15 Compatibility

Composer 8.0 is now fully compatible with Visual LANSA EPC150070, allowing you to write your own custom activity processes and utilize the new LANSA runtime to build activities specific to your needs. With the LANSA version 15 runtime behind Composer 8.0, you now have greater flexibility to create customized activities that enhance your workflows and suit your needs.



Examples include

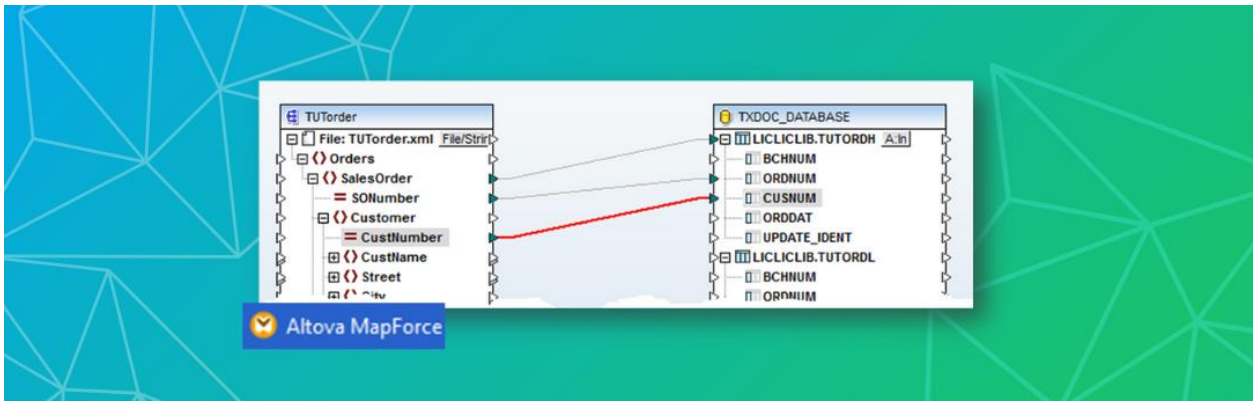
- Crafting a custom activity to split a sting using regular expressions utilising PRIM_REGEX,
- Parsing and reading a json document utilising PRIM_JSON and PRIM_IOC.
- Creating A JSON Web Token for Http verification, utilising #PRIM_SRVM.HttpJsonWebTokenSignature and #PRIM_Capi.PrivateKey

```
* ------------------------------------------------------------------------
* Global variables and components
* ------------------------------------------------------------------------
* <replace the following lines with code that performs the function of your activity>
Define_Com Class(#PRIM_REGX.MatchAll) Name(#Match)
Define_Com Class(#PRIM_REGX.MatchCollection) Name(#Matches) Reference(*DYNAMIC)
```

Altova Mapforce 2024R2 Integration

With Mapforce 2024R2 Shipped with Composer, new mapping possibilities are unlocked, including:

- Updated support for DB2 V7r4 databases.
- Expanded support for a greater number of database versions and types.
- Additional EDI versions for improved data exchange.
- Support for JSON mappings.
- Support for JSON Schemas.
- Reading and writing for more customisation.
- EDI to XML conversion for streamlined data transformation.
- Data extraction from PDFs for enhanced document processing.
- New sleep function for improved workflow automation.



Windows 11 support

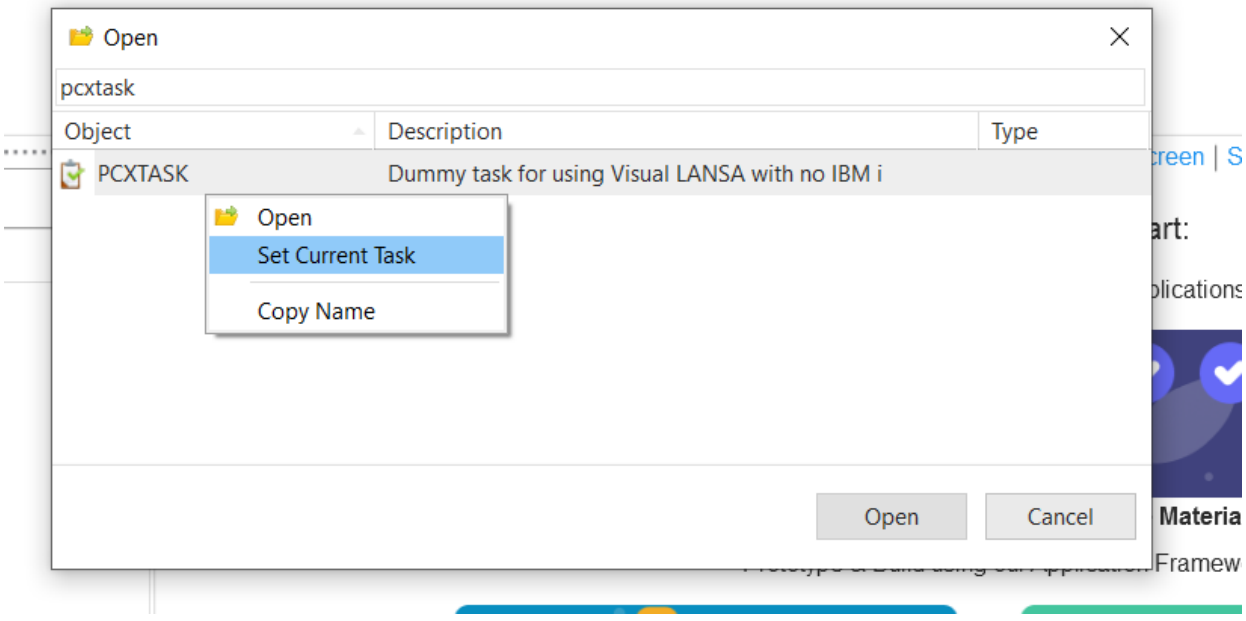
Stay ahead with full Windows 11 compatibility, providing a stable and efficient experience on Microsoft's Windows 11 operating system.

Did you know?

Set Current Task

If you work with Task Tracking, the list of tasks can be enormous. If you are already active as a developer in the Visual LANSA environment, it is possible to switch tasks. You can then go to the list of tasks, select the desired task and use the right mouse button to select the option "Set Current Task".

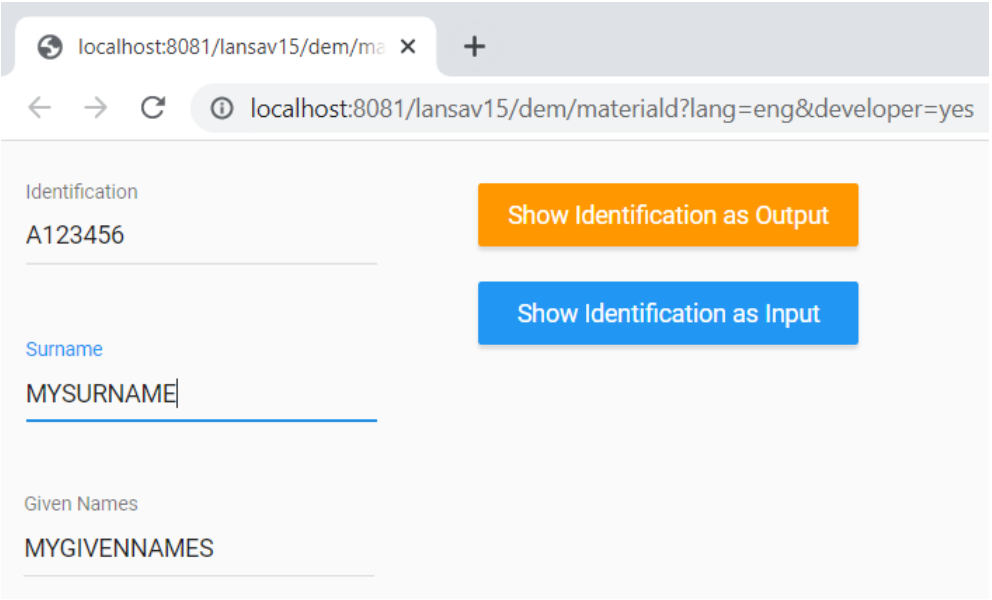
However, if you know which task you want to switch to, you can also use CTRL-O (Open) to enter the name of the task there and use your right mouse button to switch tasks.



Material Design Output Field

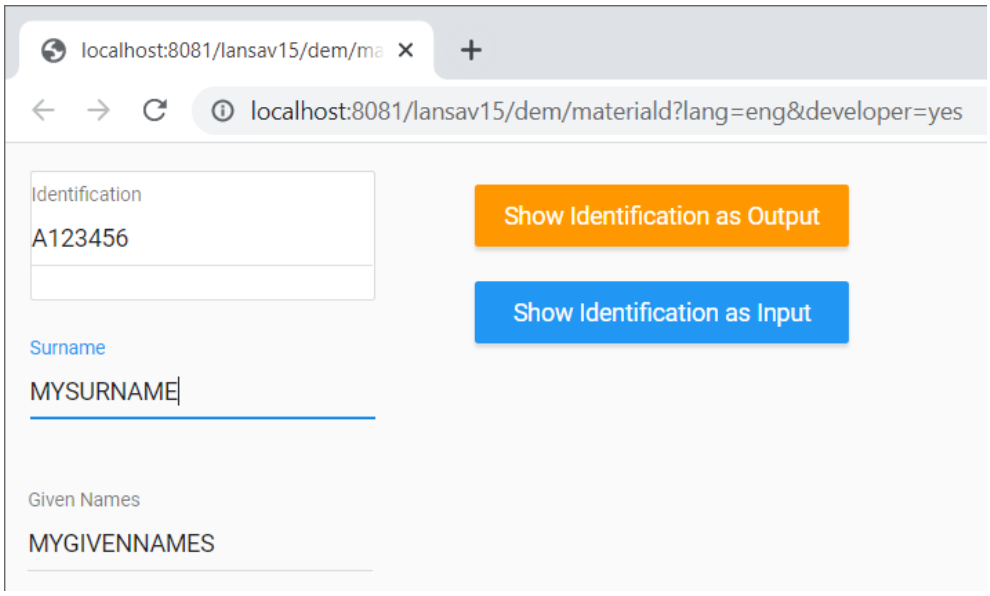
Visual LANSA Web uses Google's Material Design to display field values.

Fields that cannot be changed can be given the ReadOnly(True) property, but end users will not see that it is an output field.



This can be easily solved by changing the ThemeDrawStyle property of the field.

Below is an example of how this can be visualized, but of course any other appearance is also possible.



You can copy/paste source below to test this yourself (you have to create a new WebPage component):

```
Begin_Com Role(*EXTENDS #PRIM_WEB) Theme(#SYS_THEME<MaterialDesignBlue>)
Height(456) Width(560)

Define_Com Class(#xEmployeeIdentification.EditField) Name(#xEmployeeIdentification)
Displayposition(1) Left(14) Parent(#COM_OWNER) Tabposition(1) Top(16)
Define_Com Class(#xEmployeeSurname.EditField) Name(#xEmployeeSurname)
Displayposition(2) Left(14) Parent(#COM_OWNER) Tabposition(2) Top(106)
Define_Com Class(#xEmployeeGivenNames.EditField) Name(#xEmployeeGivenNames)
Displayposition(3) Left(13) Parent(#COM_OWNER) Tabposition(3) Top(194)
Define_Com Class(#PRIM_MD.RaisedButton) Name(#ButtonOutput) Caption('Show
Identification as Output') Displayposition(4) Left(272) Parent(#COM_OWNER)
Tabposition(4) Themedrawstyle('MediumAccent') Top(24) Width(217)
Define_Com Class(#PRIM_MD.RaisedButton) Name(#ButtonInput) Caption('Show
Identification as Input') Displayposition(5) Left(272) Parent(#COM_OWNER) Tabposition(5)
Themedrawstyle('MediumTitle') Top(80) Width(217)


Evtroutine Handling(#Com_owner.Initialize)
#xEmployeeIdentification := A123456
#xEmployeeSurname := MySurname
#xEmployeeGivenNames := MyGivenNames
Endroutine

Evtroutine Handling(#ButtonOutput.Click)
#xEmployeeIdentification.Themedrawstyle := ('Back(236,239,241,1)+Card')
Endroutine

Evtroutine Handling(#ButtonInput.Click)
#xEmployeeIdentification.Themedrawstyle := *null
Endroutine

End_Com
```

Of course it is also possible (and perhaps better) to include something like this in a Theme component.

LANSA BI 9.14

Power Up Your Applications with LANSA BI 9.14
For IBM i users, LANSA BI 9.14 is more than an upgrade. It's a modernization catalyst. This release helps you embed more business intelligence components into your core applications, replacing disconnected reporting tools with integrated, real-time insights.

What's new in LANSA BI 9.14?

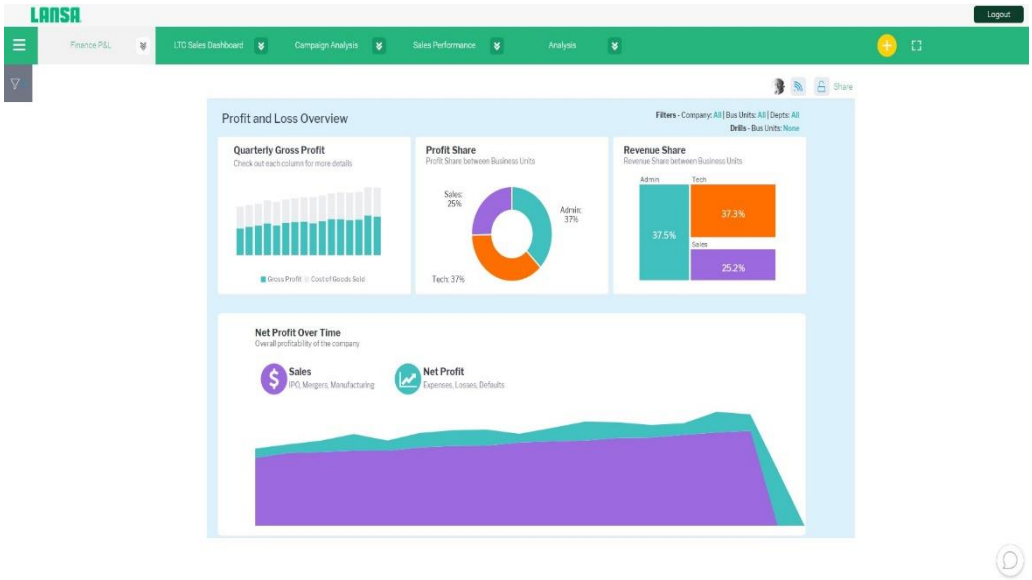
Native IBM i V7R4 / V7R5 Compatibility
LANSA BI 9.14 is built to run directly on IBM i with seamless access to Db2 data, eliminating the need for external reporting systems or complex data transfers. This allows you to deliver insights faster, reduce latency, and simplify your architecture.

Expanded REST APIs for Embedded Business Intelligence
Embedding business intelligence isn't just about displaying charts — it's about transforming how your users work. LANSA BI 9.14 levels up your integration options with powerful new REST APIs built for automation and flexibility.

What's new:

- **/content**
  Browse all user-accessible views, reports, and dashboards to build dynamic, personalized interfaces.
- **/dashboards, /presentations, /themes**
  Access and manage content metadata to craft custom embedded experiences.
- **/health**
  Monitor deployment status with both authenticated and unauthenticated endpoints.
- **/users and /import-export**
  Simplify user updates and view content dependencies during import.
- **Cached filter refreshes**
  Trigger updates on dashboards and views using REST (previously limited to SOAP).
- **New Integration API**
  Easily retrieve user and org context for plugin development, custom headers, and internal apps.

These enhancements give developers the tools to automate client onboarding in multi-tenant environments, build interactive dashboards inside LANSA applications, and streamline business intelligence workflows across the board.



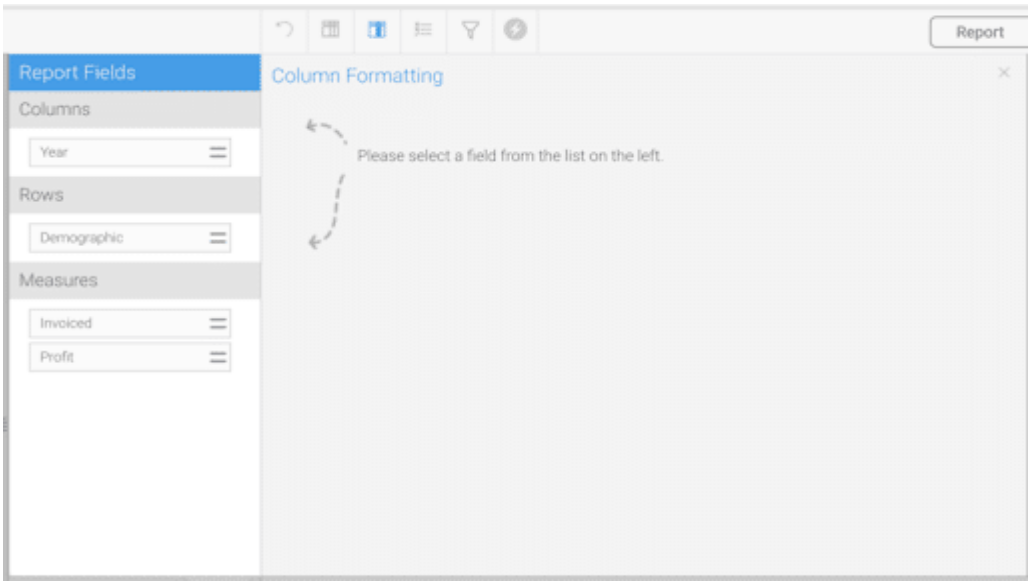
Quick-Start LANSA Application for BI Integration

To help you get started quickly, LANSA BI 9.14 includes a sample Visual LANSA application that shows how to embed and interact with reports, dashboards, and content using the new APIs.
This ready-to-use application can be tailored to visualize your company data in a few easy steps and accelerate your time-to-market.

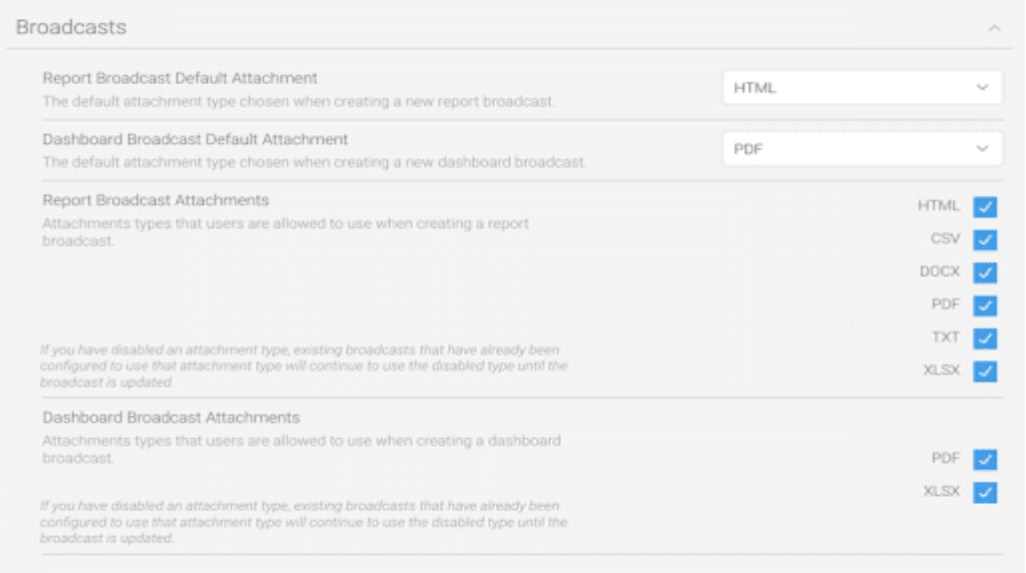
Enhanced Report Builder & UX Updates

LANSA BI 9.14 introduces important upgrades that enhance both how data is calculated and how reports are styled:

- **Drag-and-drop column reordering**
  Reorganize columns quickly from the Column Formatting page — especially helpful for large, complex reports.



- **Calculated Totals in charts**
  Recalculate values like ratios (e.g. Return on Equity) at total/subtotal levels, ensuring accurate KPIs beyond traditional aggregations.
- **Visual styling improvements**
  Customize elements such as chart label line colors to create more polished, readable reports aligned with user preferences or branding.
  These enhancements make it easier for teams to deliver precise, insight-rich reports that meet complex requirements, without adding friction to the build process.
- **Unified UI Popups**
  Cleaner, consistent popups for sharing, editing, creating folders, and more.
- **Folder Access Management**
  Resize columns, filter by People/Groups/Clients, and always see Save/Cancel buttons while scrolling.
- **Attachment Controls**
  Separate roles for TXT and CSV exports, and more granular control over broadcast attachment types.



- **Image Manager Enhancements**
  Paste images from your clipboard directly into dashboards, stories, or presentations.
- **Custom Avatars**
  User initials now appear by default, and background colors can be personalized using a color picker or hex codes.

SQL WHERE and NVARCHAR Fields on iSeries

The LANSA demo table xContacts contains some NVARCHAR fields:

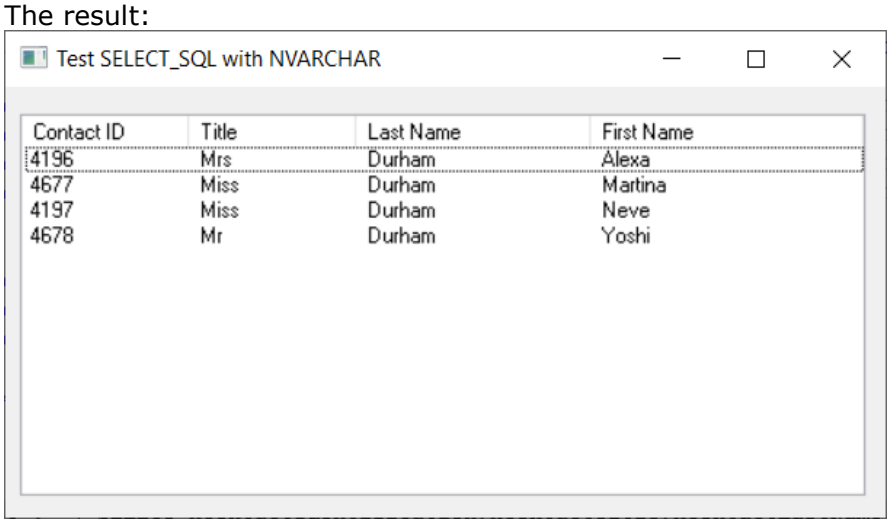
	Identifier	Description	Details
xContacts	XCONTACTS	Contacts	DC@DEMOLIB
xContactsByName	XCONTACTV1	Contacts By Last and First Name	DC@DEMOLIB
xContactIdentification	XCONID	Contact ID	Integer(4)
xContactTitle	XCONTACTT	Title	NChar(40)
xContactLastName	XCONLAME	Last Name	NVarChar(50)
xContactFirstName	XCONFNAME	First Name	NVarChar(50)
xContactGender	XCONTACTG	Gender	Alphanumeric(15)
xContactEmailAddress	XCONEMAIL	Email Address	NVarChar(200)

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientwidth(484) Clientheight(237) Componentversion(2) Left(696) Top(220)

  Define_Com Class(#PRIM_LTVW) Name(#ListView1) Componentversion(2) Displayposition(1) Fullrowselect(True)
       Keyboardpositioning(SortColumn) Left(8) Parent(#COM_OWNER) Showsortarrow(True) Tabposition(1) Top(15) Height(210) Width(465)
    Define_Com Class(#PRIM_LVCL) Name(#LVCL1) Displayposition(1) Parent(#ListView1) Source(#xContactIdentification)
    Define_Com Class(#PRIM_LVCL) Name(#LVCL2) Displayposition(2) Parent(#ListView1) Source(#xContactTitle)
    Define_Com Class(#PRIM_LVCL) Name(#LVCL3) Displayposition(3) Parent(#ListView1) Source(#xContactLastName) Width(28)
    Define_Com Class(#PRIM_LVCL) Name(#LVCL4) Displayposition(4) Parent(#ListView1) Source(#xContactFirstName) Widthtype(Remainder)

  Define_Com Class(#prim_dc.UnicodeString) Name(#u_Where)
    Define_Com Class(#prim_dc.UnicodeString) Name(#u_Select)
    Define_Com Class(#prim_dc.UnicodeString) Name(#u_Query)

  Evtroutine Handling(#com_owner.CreateInstance)

     Set Com(#com_owner) Caption(*component_desc)

     #u_Select := ("SELECT xContactIdentification,xContactTitle,xContactLastName,xContactFirstName FROM XDEMOLIB.XCONTACTS")
     #u_Where := ("xContactLastName = 'Durham'")
     #u_Query := #u_Select + " where " + #u_Where

    Select_Sql Fields(#xContactIdentification #xContactTitle #xContactLastName #xContactFirstName) From_Files((xContacts))
        Using(#u_Query.asNativeString)
      Add_Entry To_List(#ListView1)
    Endselect

  Endroutine

End_Com
```

In Windows you can see that in the select we can use the SQL select longnames without any problem:

```
#u_Select := ("SELECT xContactIdentification,xContactTitle,xContactLastName,xContactFirstName
FROM XDEMOLIB.XCONTACTS")
#u_Where := ("xContactLastName = 'Durham'")
#u_Query := #u_Select + " where " + #u_Where

Select_Sql Fields(#xContactIdentification #xContactTitle #xContactLastName #xContactFirstName)
From_Files((xContacts)) Using(#u_Query.asNativeString)
Add_Entry To_List(#ListView1)
Endselect
```

The result:



The SQL environment on the iSeries however reacts different.

If you use instruction below for example you can see that it has problems with the field longname:

```
select * from dc@demolib/xcontacts where xContactLastName = 'Durham'
Column or global variable XCONTACTLASTNAME not found.
```

This can be solved by placing quotes around the field name:
select * from dc@demolib/xcontacts where "xContactLastName" = 'Durham'

Or you can use the Identifier name (XCONLAME).

In both situations you will get a strange error, which is related to the WHERE clause:

```
SELECT * FROM dc@demolib/xcontacts WHERE XCONLAME = 'Durham'
Character conversion between CCSID 1200 and CCSID 65535 not valid.
```

The reason is the NVARCHAR field type.

On the iSeries, you have to specify the length parameter n with NVARCHAR column types, because if you don't it's equivalent to n=1.

So for a correct syntax on the iSeries, you have to put the n character before the value to check, like this:

```
SELECT * FROM dc@demolib/xcontacts WHERE XCONLAME = n'Durham'
Uitvoering SELECT-instructie voltooid.
```

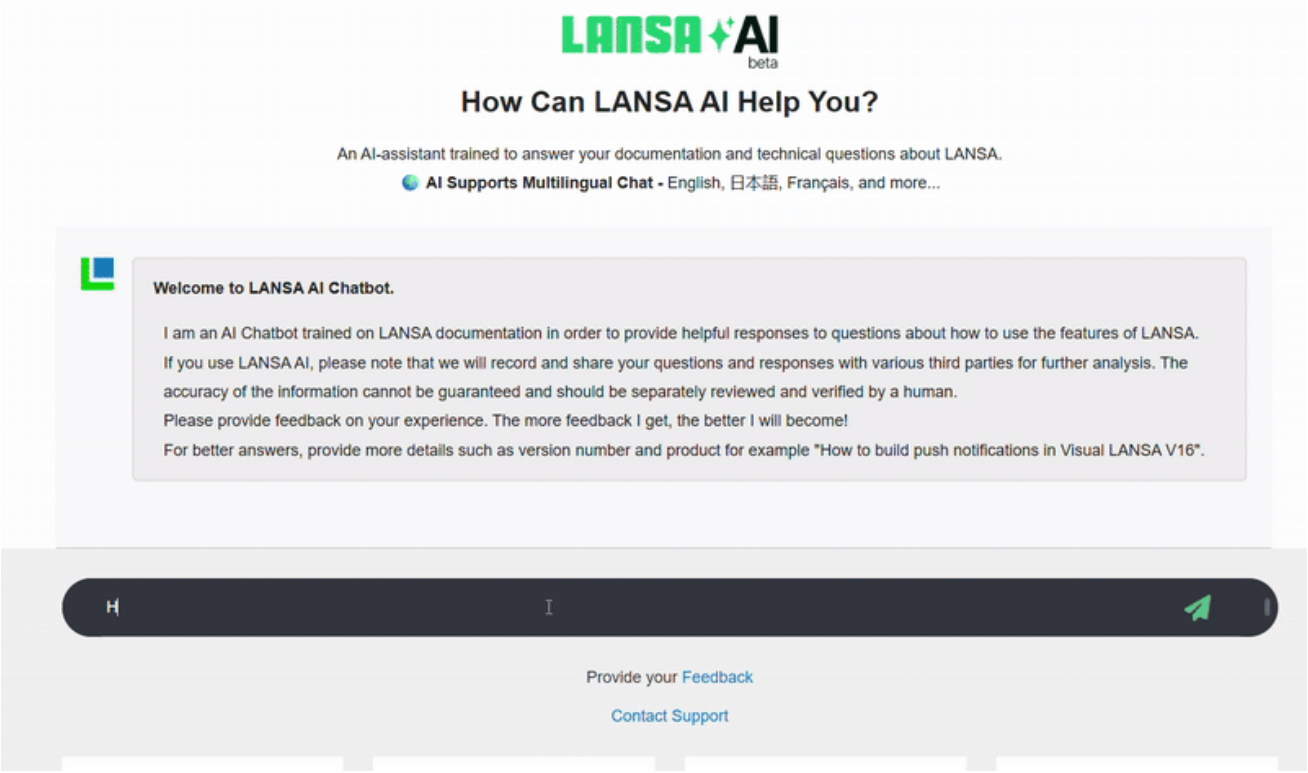
SELECT * FROM dc@demolib/xcontacts WHERE XCONLAME = n'Durham'

A correct SQL syntax in LANSA via SELECT_SQL could therefore look like this:

```
00001       Function Options(*DIRECT) Rcv_List(#wlist)
00002
00003       Def_List Name(#wlist) Fields(#XCONID #XCONTACTT #XCONLAME #XCONFNAME) Type(*WORKING) Entrys(*MAX)
00004
00005       Define_Com Class(#prim_dc.UnicodeString) Name(#u_Where)
00006       Define_Com Class(#prim_dc.UnicodeString) Name(#u_Select)
00007       Define_Com Class(#prim_dc.UnicodeString) Name(#u_Query)
00008
00009       #u_Select := ("SELECT XCONID,XCONTACTT,XCONLAME,XCONFNAME FROM DC@DEMOLIB/XCONTACTS")
00010       #u_Where := ("XCONLAME = n'Durham'")
00011       #u_Query := #u_Select + " where " + #u_Where
00012
00013       Select_Sql Fields(#XCONID #XCONTACTT #XCONLAME #XCONFNAME) From_Files((XCONTACTS))
                Using(#u_Query.AsNativeString)
00014         Add_Entry To_List(#wlist)
00015       Endselect
```

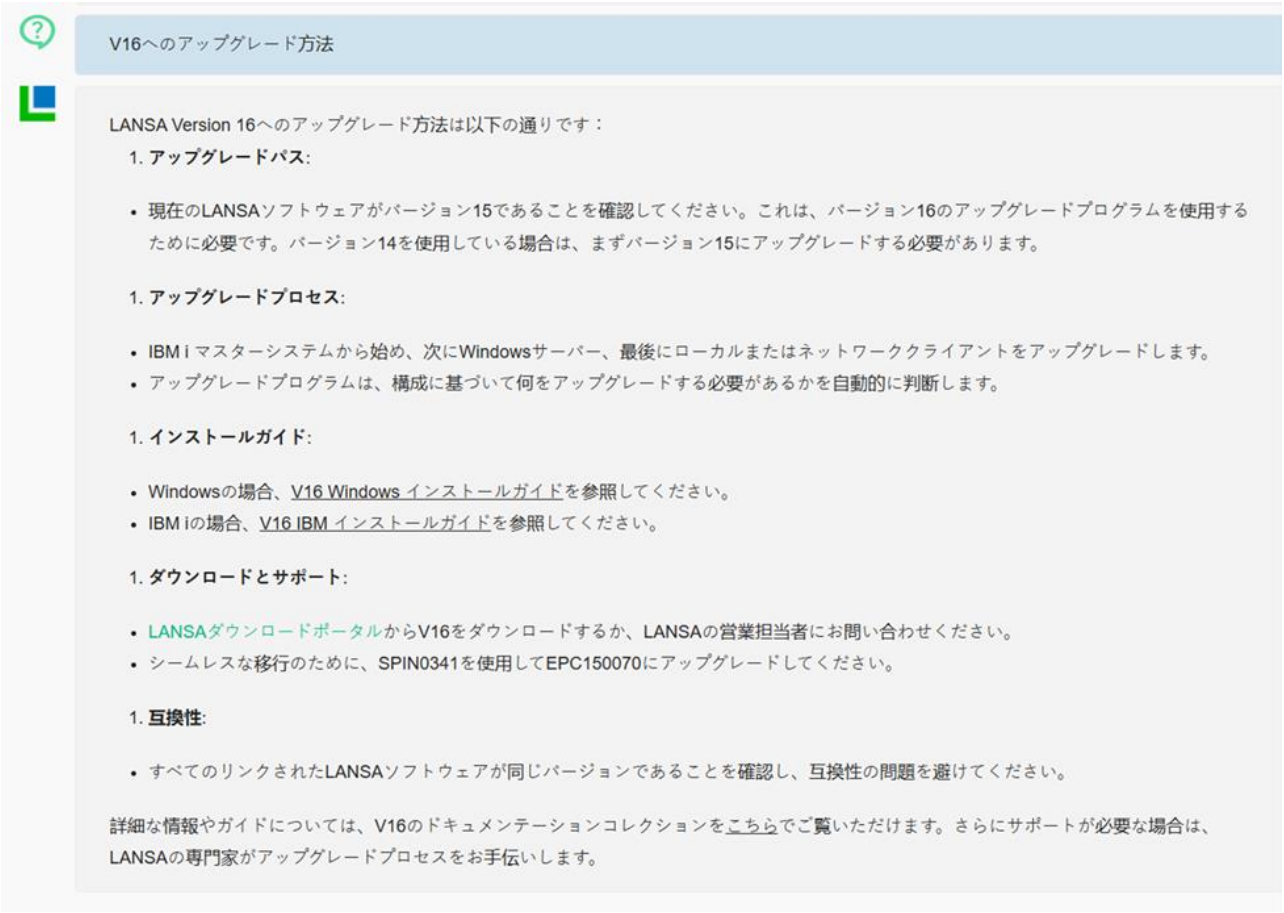
Introducing the LANSA AI Chatbot

Ask, learn, and build code smarter, and solve faster in any user language, any time, with LANSA AI Chatbot as your new Visual LANSA V16 assistant.



The new LANSA AI Chabot (Beta) is available on lansa.com/support from 8 July 2025 onwards.

- Get answers to your documentation and technical queries through a fast search of LANSA documentation for LANSA products.
- Instant access to LANSA documentation—ask questions instead of searching.
- Exclusively trained and Built-in understanding of LANSA product names, functions, and versioning (e.g., Visual LANSA V16).
- Ask technical "how-to" questions and receive clear, structured guidance.
- Context-aware answers based on real product documentation.
- Works in your browser—no download, no setup.
- Continuous improvement based on user feedback and interaction analytics.
- Automatically redirects questions on pricing, premium support, and sales to the appropriate LANSA contact channels
- AI Supports Multilingual Chat - English, 日本語, Français, and more…



V16へのアップグレード方法

LANSA Version 16へのアップグレード方法は以下の通りです：

1. **アップグレードパス**:

- 現在のLANSAソフトウェアがバージョン15であることを確認してください。これは、バージョン16のアップグレードプログラムを使用するために必要です。バージョン14を使用している場合は、まずバージョン15にアップグレードする必要があります。

1. **アップグレードプロセス**:

- IBM i マスターシステムから始め、次にWindowsサーバー、最後にローカルまたはネットワーククライアントをアップグレードします。
- アップグレードプログラムは、構成に基づいて何をアップグレードする必要があるかを自動的に判断します。

1. **インストールガイド**:

- Windowsの場合、V16 Windows インストールガイドを参照してください。
- IBM iの場合、V16 IBM インストールガイドを参照してください。

1. **ダウンロードとサポート**:

- LANSAダウンロードポータルからV16をダウンロードするか、LANSAの営業担当者にお問い合わせください。
- シームレスな移行のために、SPIN0341を使用してEPC150070にアップグレードしてください。

1. **互換性**:

- すべてのリンクされたLANSAソフトウェアが同じバージョンであることを確認し、互換性の問題を避けてください。

詳細な情報やガイドについては、V16のドキュメンテーションコレクションをこちらでご覧いただけます。さらにサポートが必要な場合は、LANSAの専門家がアップグレードプロセスをお手伝いします。

Why use LANSA AI Chatbot?

- Easy to use and get answers fast without digging through pages of documentation.
- Multilingual—supports global teams and up to 100 languages.
- Made for developers, consultants, and support staff - Empowers global teams with instant access to technical answers 24/7.
- Understands technical context like “push notifications in V16”.
- Available 24/7—no waiting on time zones or office hours.
- Continuously gets smarter with your feedback.
- No extensions or plugins required—just start typing your question in LANSA AI Chabot in LANSA support webpage.

End of Support for Visual LANSA 14 and Earlier

With the launch of V16, support for V14 and earlier ends today, except where current agreements apply. Talk to your Account Manager now to explore your options and plan your upgrade.

By moving to V16, you gain access to all the new features, ensure your applications continue to perform at their best, and minimize risks to your business continuity.