

What's New in LANSA Version 13 Service Pack 2?

The second Service Pack for LANSA Version 13 includes new features and some enhancements.

[Web Application Module Enhancements](#)

Various WAM enhancements are introduced in SP2.

[Windows 64-bit Support](#)

Visual LANSA now supports the generation and compilation of both Windows 32-bit and 64-bit applications.

[IBM i User Profile Handling](#)

IBM i user profiles can now be validated, and IBM i passwords can be changed using the IBM special APIs with interfaces to RDML and LANSA Open.

[SuperServer Enhancements](#)

SuperServer connections can now be made between all supported platforms: IBM i, Windows and Linux.

[IDE Enhancements for IBM i Administrators](#)

Export lists can be generated automatically when checking in or delivering objects to an IBM i server, and objects can be refreshed selectively in the Repository.

Before deploying your Version 13 applications, please take time to thoroughly understand the new [MSI Deployment](#) mechanism.

Note that Version 13.0 iSeries Exports cannot be imported to V13.0 SP1 or later versions.

This document also contains [What's New in LANSA Version 13?](#) and [What's New in LANSA Version 13 SP1?](#)

Edition Date May 30, 2014

© 2014 LANSA

Web Application Module Enhancements

LANSA Version 13 Service Pack 2 introduces many new features and enhancements for WAMs:

[jQuery Mobile WAM Enhancements](#)

[Support for file uploads to a webroutine](#)

[New XHTML Weblets](#)

[Upgraded Third-Party Libraries](#)

[WAM Editor Enhancements](#)

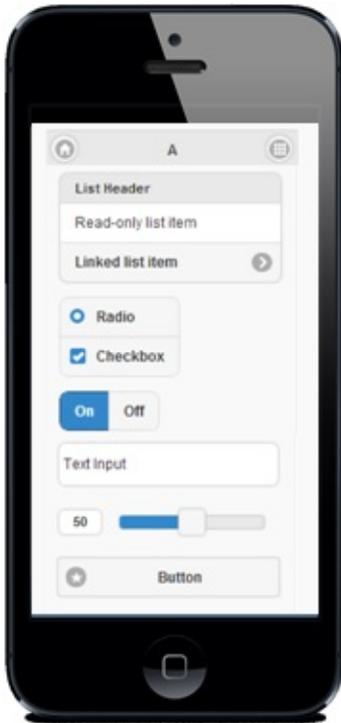
jQuery Mobile WAM Enhancements

Commonly used jQuery Mobile weblets have been made easier to use and various new weblets have been introduced.

[Easier to Design jQuery Mobile WAMs](#)

[New jQuery Mobile Weblets](#)

[Other improvements to jQuery Mobile](#)



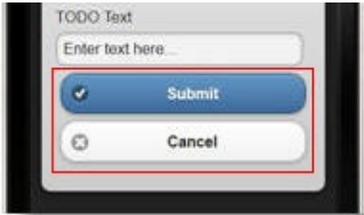
Note that the jQuery Mobile look and feel has changed:

- New Flat UI
- Two themes: Light and Dark
- Themeroles based theming
- Widgets can be pre-rendered for better performance.

Easier to Design jQuery Mobile WAMs

Simplified default versions of a number of jQuery Mobile Weblets have been introduced for ease of use.

These include the button (std_button_s1):



And anchor (std_anchor_s1) weblets:

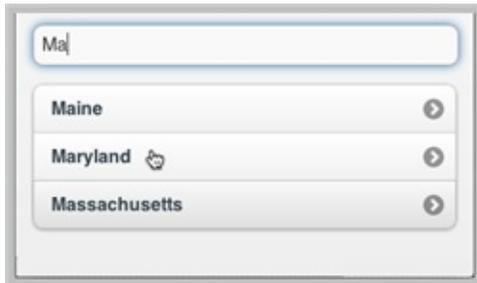


Use the previous versions of these weblets (std_button_v2 and std_anchor_v2) only if you need to add content to them.

New jQuery Mobile Weblets

New jQuery Mobile weblets are available:

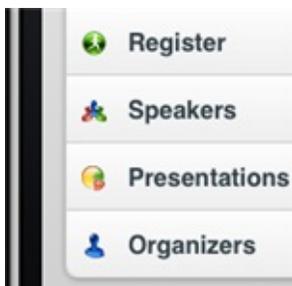
Autocomplete



The Autocomplete weblet provides suggestions while you type into the field.

The suggestions are provided by a webroutine using Ajax.

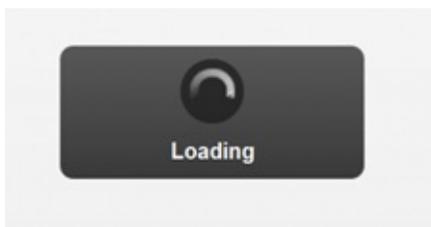
Image



The Image weblet displays an image.

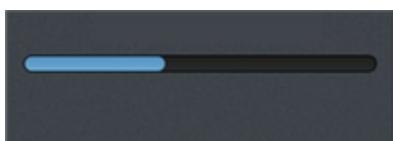
The weblet has an option to load the image only when it comes into view, which helps render the page faster.

Loader



The Loader displays a small loading overlay when jQuery Mobile loads in content via AJAX, or when you want to perform an action that momentarily blocks user interaction.

Progress bar



Progress bar displays the status of a determinate process.

It can also be used to display a value as a percentage of its maximum value.

Other improvements to jQuery Mobile

Input Box Weblet

The Input Box weblet now supports input type="number" to bring the correct keyboard to mobile devices.



Generated Lists for jQuery Mobile

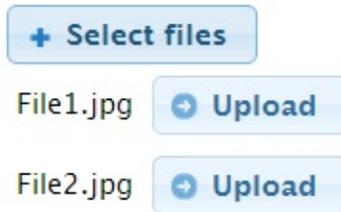
Generated lists for jQuery Mobile are now similar in structure to the lists generated for Technology Service XHTML. You have access to columns by variable name.

Previously the XHTML and jQuery Mobile list methods were different, with the jQuery Mobile's method giving more flexibility in adding content to the list entry at the expense of making it less easy to use in the WAM Editor.

You can still use the more flexible and complex method by using the jQuery Mobile `std_html_list` weblet.

Support for File Uploads to a Webroutine

You can use the file upload weblet to select files to upload to the application server (into a temporary directory). The webroutine that receives the file upload can then manipulate the uploaded files as required.



For more information, see the description of the [XHTML File Upload \(std_fileupload\) weblet](#) and for the [jQuery Mobile File Upload \(std_fileupload\) weblet](#).

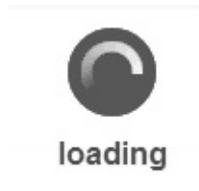
New XHTML Weblets

New XHTML Mobile weblets are available:

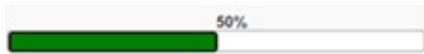


The Image weblet displays an image.

The weblet has an option to load the image only when it comes into view, which helps render the page faster.



The Loader displays a small loading overlay when jQuery Mobile loads in content via AJAX, or when you want to perform an action that momentarily blocks user interaction.



Progress bar displays the status of a determinate process.

It can also be used to display a value as a percentage of its maximum value.

Upgraded Third-Party Libraries

Third-party libraries have been upgraded:



jQuery Core 1.9.1



jQuery UI 1.10.3



jQuery Mobile 1.4.2



jQuery Timepicker Plugin 1.4.3



CKEditor 4.2.1



Mobiscroll 2.9.5

WAM Editor Enhancements

The WAM Editor now inspects the design of a web routine for use of deprecated weblinks and lets the user know if they are using deprecated weblinks. This test is done when a web routine's design is opened in the WAM Editor.

Windows 64-bit Support



Visual LANSA now supports the generation and compilation of both Windows 32-bit and 64-bit LANSA applications.

- A 64-bit Visual LANSA runtime is provided in addition to the 32-bit runtime
- 64-bit deployments are supported in addition to 32-bit deployments

[When Should Windows 64-bit Support be Enabled?](#)

[Installation Considerations](#)

[Programming Considerations](#)

[32-bit and 64-bit Applications Accessing the Same Database](#)

[Notable Environmental Differences](#)

When Should Windows 64-bit Support be Enabled?

We recommended that Windows 64-bit support is only enabled when there is corporate requirement for it.

Windows 64-bit support should only be installed on a Build machine, not developer machines.

Drawbacks

Using Windows 64-bit support has some drawbacks:

- You must obtain your own 64-bit compiler, either Visual Studio 2010 Professional (or later) or Visual Studio 2012 Express for Desktop (or later).
- Compile times are longer because both 32-bit and 64-bit DLLs are always built.
- Functions which use DISPLAY, REQUEST or POPUP commands will fail to compile also in 32-bit DLLs.

Features not Supported

There are LANSAs features that do not function or are not supported in 64-bit applications:

- Graphics Server
- Web Functions
- ZIP and specialized LANSAs Built In Functions (BIFs)
- Explorer Component AutoRefresh Property

Installation Considerations

It is presumed that 64-bit support is only enabled on the Build machine and that developers do not enable it. As a consequence, when 64-bit support is enabled both the 32-bit and 64-bit compiles are performed and both MSI packages are built.

You cannot compile a function which contains DISPLAY, REQUEST and POPUP commands - even the 32-bit compile will fail. This is why it is better not to enable 64-bit support on Developer's machines. If developers need to work on both RDML functions and 64-bit applications, two systems which use the same repository can be installed on their machines

Compiler Installation

If a supported compiler is not installed before LANSAs, the LANSAs-shipped compiler is installed and enabled. If you install a 64-bit compiler later, you need to change this registry entry to disable the LANSAs-shipped compiler:

On a 64-bit PC HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\LANSAs
 to 0.

On a 32-bit PC HKEY_LOCAL_MACHINE\SOFTWARE\LANSAs\MicrosoftCom
 bit PC

If Visual Studio is installed before LANSAs, it is used as the compiler.

If the version of Visual Studio installed does not support 64-bit compiling, install a version that does. LANSAs will detect it when it is next started.

.

Programming Considerations

There are no increases in the maximum size of any LANSAs feature because the limits are considered sufficient. This ensures greater compatibility between 32-bit and 64-bit applications. For example:

- The maximum size of an RDMLX List is still 2 billion rows, with each entry being 2 billion bytes long.
- The Built In Functions SND_TO_DATA_QUEUE and RCV_FROM_DATA_QUEUE may be used interchangeably.
- Job Queue Emulation can use either a 32-bit or 64-bit Job Queue Monitor and jobs may be submitted from either 32-bit or 64-bit. Note that the 64-bit Job Queue Monitor will execute the submitted job as 64-bit, no matter which platform submitted the job.

PC Other Files which are loaded using a 32-bit ODBC driver will need to create a 64-bit DSN with the same name as that used to load the file, or use CONNECT_SERVER when deployed to re-direct IO to a 64-bit driver.

An ActiveX included in LANSAs RDML must be a registered 32-bit version. To execute the ActiveX, a version must be registered which is of the same processor architecture as the LANSAs runtime. That is, if the LANSAs runtime is 64-bit then the 64-bit ActiveX must be registered on the deployed PC.

32-bit and 64-bit Applications Accessing the Same Database

These considerations are particularly important when deploying an application into a production system:

- | | |
|--|--|
| Do not create mixed 32-bit and 64-bit applications | To avoid complexity, it is recommended that applications are either 32-bit or 64-bit. For example, if you use both 32-bit and 64-bit clients when using SuperServer, only use a 64-bit server. Because the clients are not directly accessing the database, there is no complication. |
| Auto-generate relative record numbers | Assign relative record numbers using auto-generation. If relative record numbers are assigned using external files, duplicates will occur unless the RPTH parameter is assigned to the same path for both 32-bit and 64-bit applications. A file that is currently using external files can be changed to use auto-generation using the Upgrade tool feature <i>Convert Files to Use Identity Column</i> . |
| Database upgraded by first system upgraded | Table upgrades are identified by comparing the previous CTD file to the new CTD file being installed. Thus only the first system upgraded should upgrade the database. This is why database upgrade defaults to off during an MSI install and why per-user installs disable database upgrade. |
| Be consistent | If an existing OAM is not there for 64-bit but is for 32-bit, and vice versa, the user needs to control which is the latest OAM. If 32-bit is the first environment to be installed, continue that way for all Upgrades and Patches. Once the 64-bit environment is at the same level, the Upgrade/Patch database change machine can be switched, but it is inadvisable. Be consistent and use one machine from the beginning. |

Notable Environmental Differences

- The system directory for 32-bit applications is of the form x_win95\x_lansa. For 64-bit applications it is x_win64\x_lansa. Therefore system variables like *SYS_DIR return a different value.
- Visual LANSA is a 32-bit application. Hence interaction between Visual Lansa and 64-bit generated DLLs cannot occur.
- 32-bit OAMs are always built because Visual LANSA requires the 32-bit OAM to unload and load the data from the table. The 64-bit build command always skips the SQL table build, presuming that 32-bit has already done it.
- The Windows Installer has a known defect which converts the Target directory in a Shortcut from c:\program files to c:\program files (x86). Nonetheless, the shortcut still works correctly as if it was c:\program files. Even if the 32-bit version of the Application is installed in c:\program files (x86), it does not get executed, it is still the 64-bit version. See the MSDN forum post [32bit MSI on 64bit OS: Converting shortcut target path of 64bit app to 32 bit Path](#).
- A similar situation occurs with Windows\system 32. The shortcut looks OK but it does not find the object. It is not valid to create a shortcut that points to this directory.

IBM i User Profile Handling

IBM i user profiles can now be validated, and IBM i passwords can be changed using the IBM special APIs with interfaces to RDML and LANSA Open.

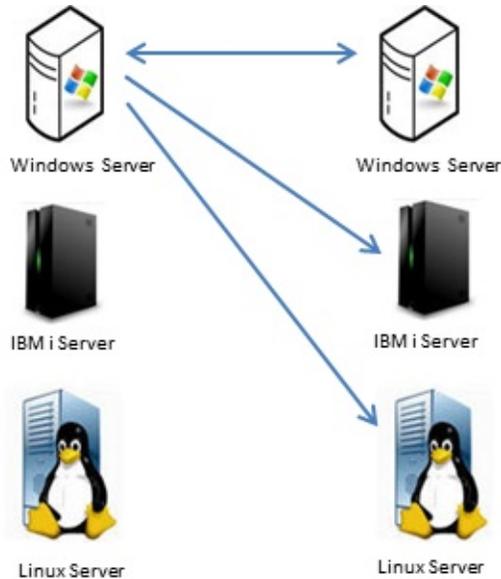
Using this functionality non-5250 users can find out when their password is going to expire, and, if the password has not expired or been disabled, to change it.

This functionality is implemented in Visual LANSA as Built-In Functions, and in LANSA Open as Lce APIs. See:

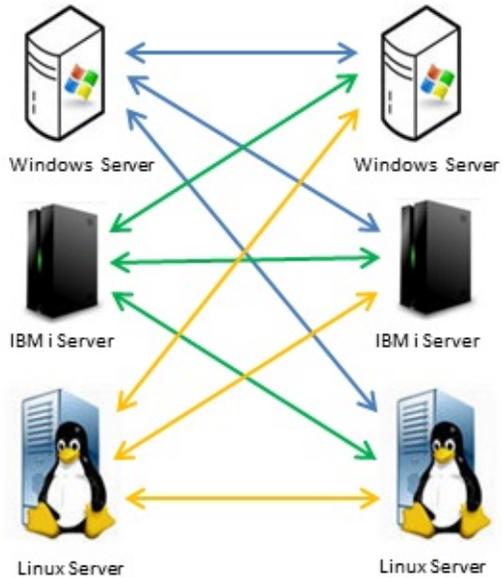
- [CHECK_IBMI_SIGNON](#) and [CHANGE_IBMI_SIGNON](#) in the Technical Reference.
- [LceGetIBMiSignon](#) and [LceSetIBMiSignon](#) in the LANSA Open Guide.

SuperServer Enhancements

Until LANSA Version 13 Service Pack 2 the possibilities of establishing a SuperServer connection between different types of servers was limited.



Now it is possible to establish a SuperServer connection between any two servers regardless of the operating system they are running.



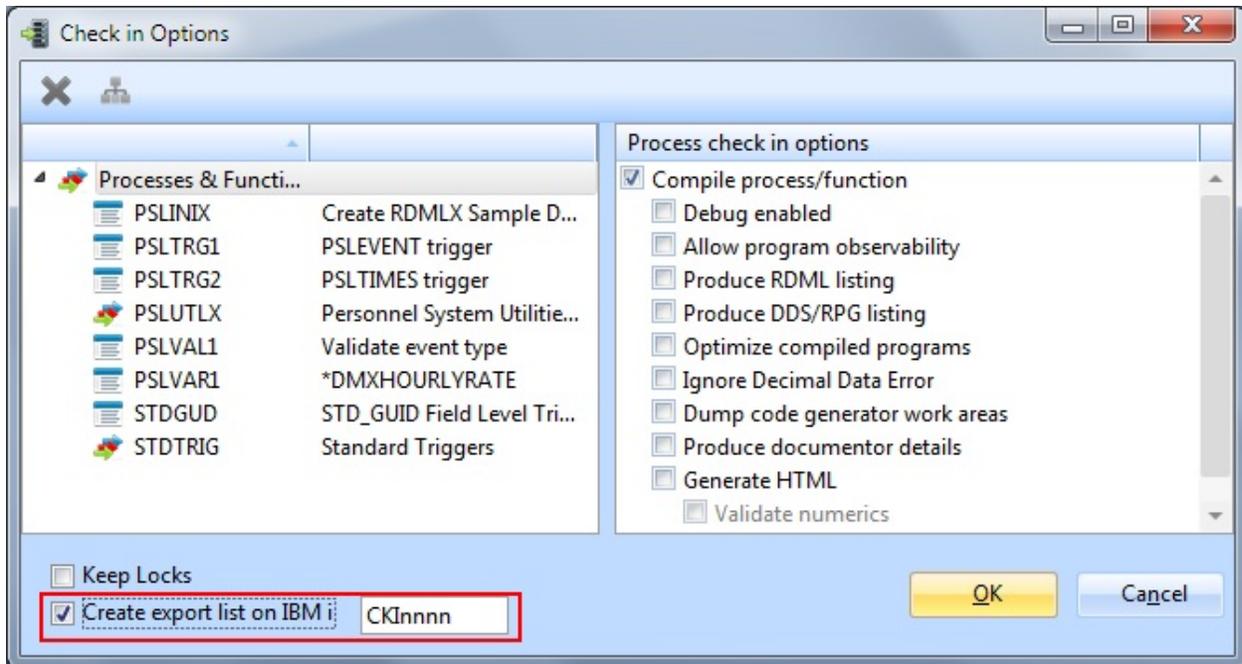
IDE Enhancements for IBM i Administrators

New Check-In Features

Refresh Selected Objects

New Check-In Features

When objects are checked in or delivered to an IBM i system, an export list containing the selected objects can be automatically generated on the IBM i:

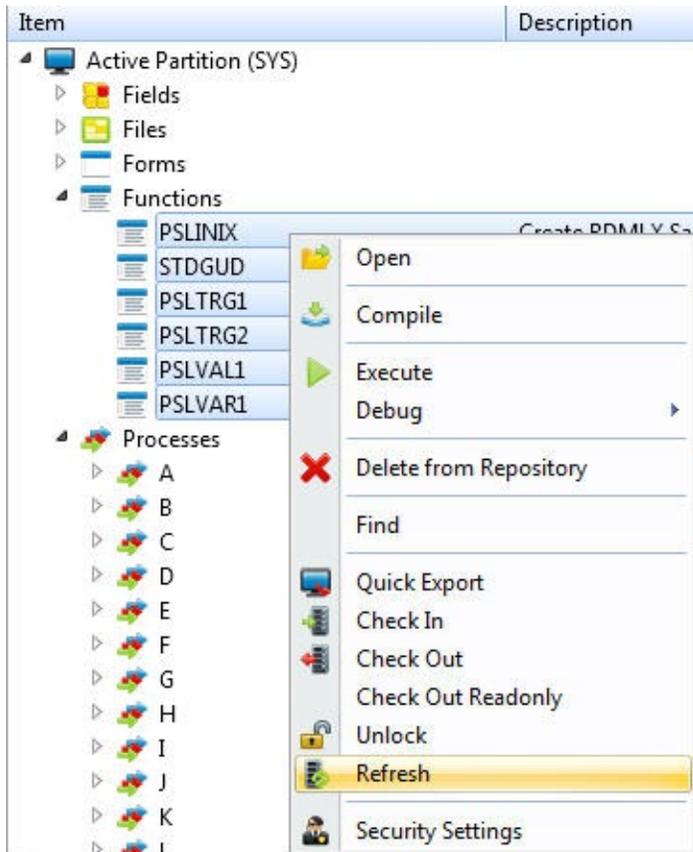


You can change the default name of the export list. If you specify the name of an existing export list, the new list will be appended to it.

For more information see [Check In Options](#) and [Deliver To Options](#) in the Visual LANSA Administrator's Guide.

Refresh Selected Objects

Instead of doing a full refresh of all the objects in the repository, you can refresh the details of selected objects which are already in the local repository by right-clicking and selecting Refresh:



Object data is downloaded from the Master in batches and then trickled into the local Repository to avoid impacting performance on the local machine. You can use F5 at any time to force the objects to be refreshed immediately.

For more information see [Refresh Master Object List](#) in the Visual LANSAs Administrator's Guide.

LANSA バージョン 13 SP1 の新機能

LANSA V13 SP1 には以下の機能拡張と新しい機能が含まれています。

Visual LANSА ユーザビリティの機能拡張

新しい**テーブル・レイアウト**がリボンより直接管理できます。

DirectX スタイルの管理機能がリボンに統合されました。

クイックアクセス ツールバーにどのコマンドを含めるかを選択できるようになりました。

エディターの**ステータスバー**をカスタマイズすることができます。

ユーザー定義コントロール (UDC) がより使いやすくなり、いくつかのサンプルデザインが使用できるようになりました。

新しい**CRUD ウィザード**は IDE インターフェイスに似たリッチクライアントアプリケーションを生成します。

新しいイメージとスタイルがリッチクライアントアプリケーションで使用できるようになりました。

ヘルプテキストの精度が改善しました。詳細は「**機能ヘルプの改善**」を参照してください。

WAM 機能拡張

ウェブルーチンは **JSON 形式の POST** をサポートします。

リクエストを送るウェブレットは、VLF 開発者が WAM イベントをセットすることができる新しいプロパティ **vf_wamevent** を持つようになりました。

WAM エディタの**元に戻す/やり直し**機能が改善されました。

ロジカル・モデラー

ロジカル・モデラーはファイルのロングネームをサポートようになりました。

LANSA Open

LANSA Open は Unicode、Boolean カラム、ロング
ネームのサポートを行なうようになりました。

V13 のアプリケーションを配布する前に、新しい MSI 配布メカニズムを
しっかり理解する時間をとってください。

V13.0 の LANSAD 側でのエクスポートは V13.1 にはインポートできな
いことに注意してください。

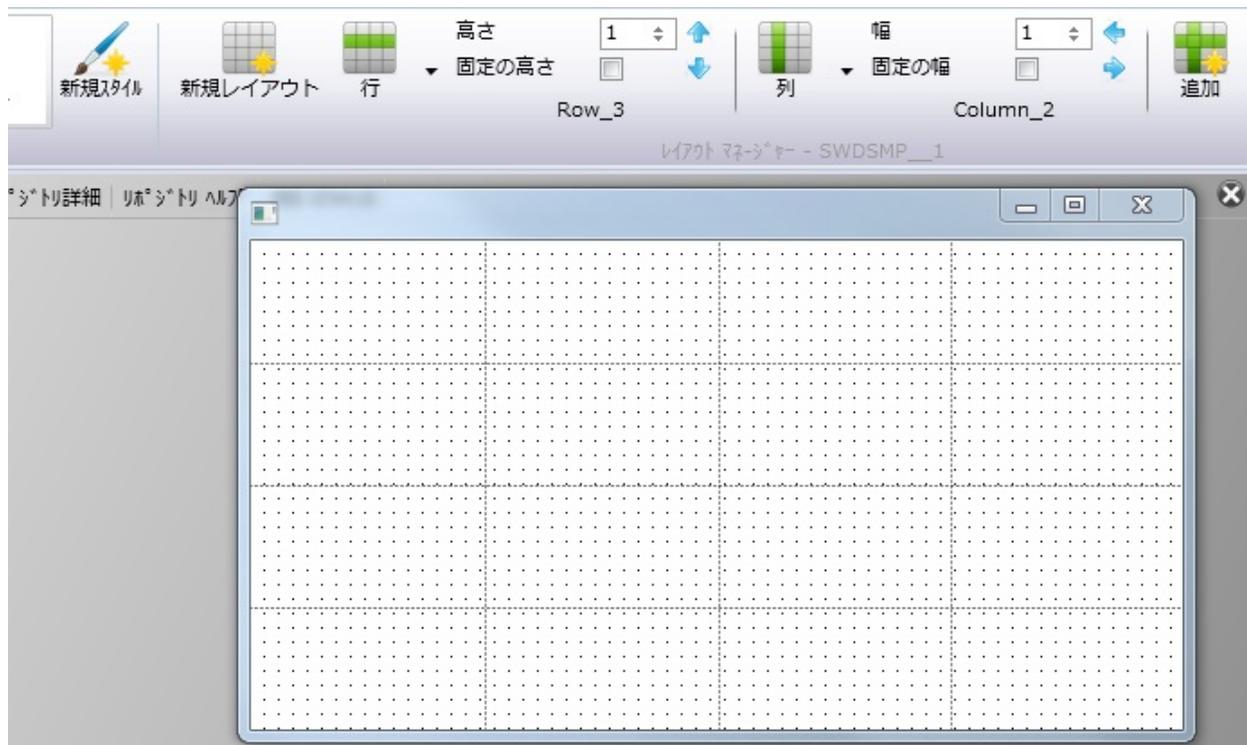
このドキュメントには、LANSA バージョン 13 の新機能も含まれま
す。

エディション日付：2013年7月1日

© 2013 LANSAD

テーブル・レイアウト

新しいテーブルレイアウトがリボンより直接管理できます。

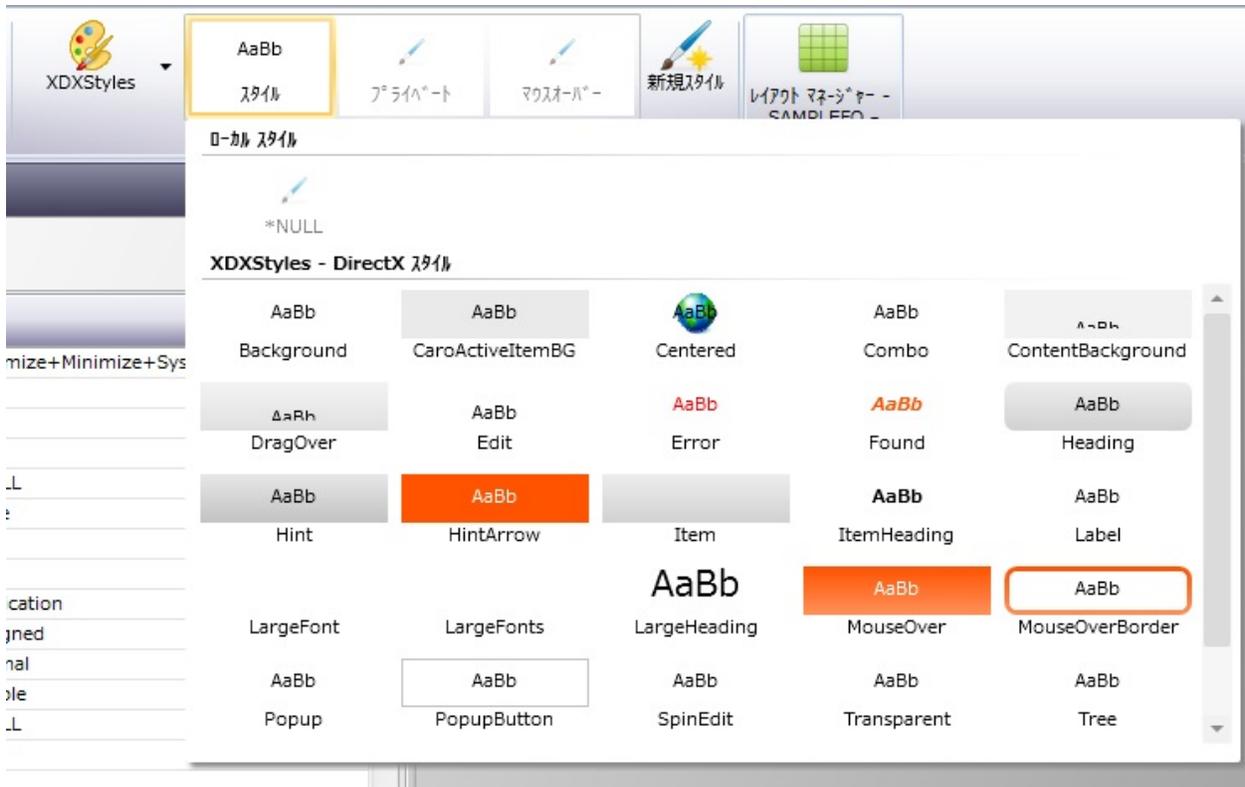


テーブルレイアウト マネージャを使用して、アプリケーション インターフェースの作成や管理を簡単に行なうことができます。従来のレイアウト マネージャは、ほとんどの場合で必要なくなりました。

[「テーブル・レイアウト」参照](#)

DirectX スタイル

DirectX スタイルはリボンから管理することができます。



[「動的スタイル」参照](#)

クイックアクセス ツールバー

クイックアクセス ツールバーにより、よく使われるコマンドに簡単にアクセスすることができます。



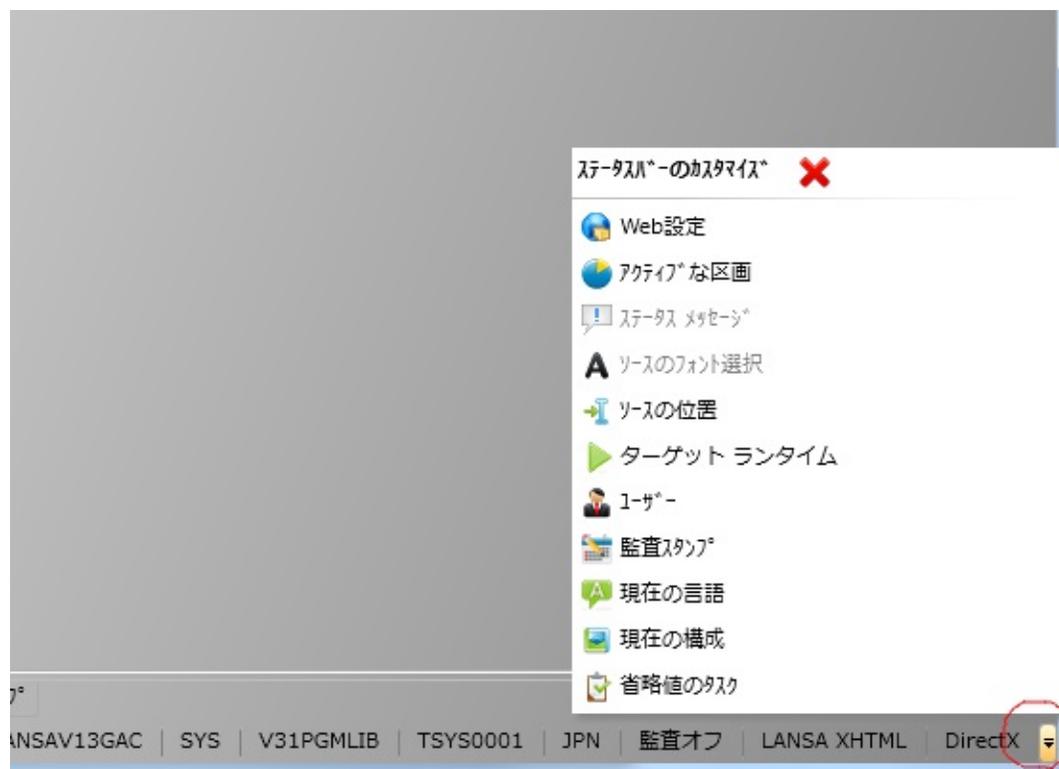
そこに含めるコマンドを選べるようになりました。

クイック アクセス ツールバーのカスタマイズ ✖

-  Web ブラウザ
-  インポート
-  エラー ログ
-  クイック エクスポート
-  コンパイル
-  システム情報
-  チェックイン
-  テキスト検索
-  デバッグ
 - ▶バージョン 12 メニューバー
-  ビルト
-  ホスト モニター
-  マスター エラー ログ
-  マスターのリフレッシュ
-  メッセージ ファイル
-  リポジトリ検索
-  開いているオブジェクト
-  開く
-  次のコンポーネント
-  実行
-  実行履歴
-  接続履歴
-  前のコンポーネント
-  配布ツール
-  閉じる

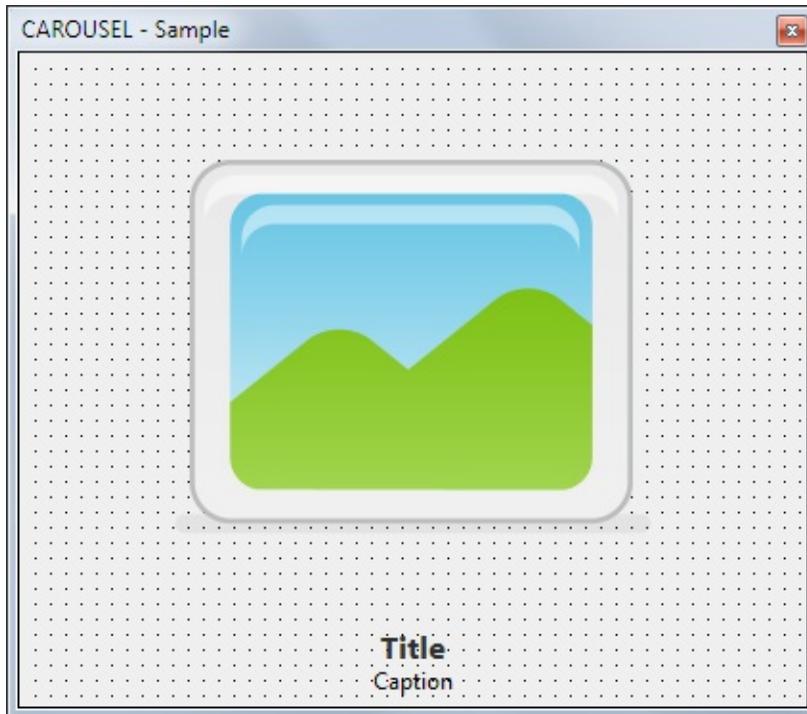
ステータス・バー

ステータス・バーに表示する情報を選択することができます。

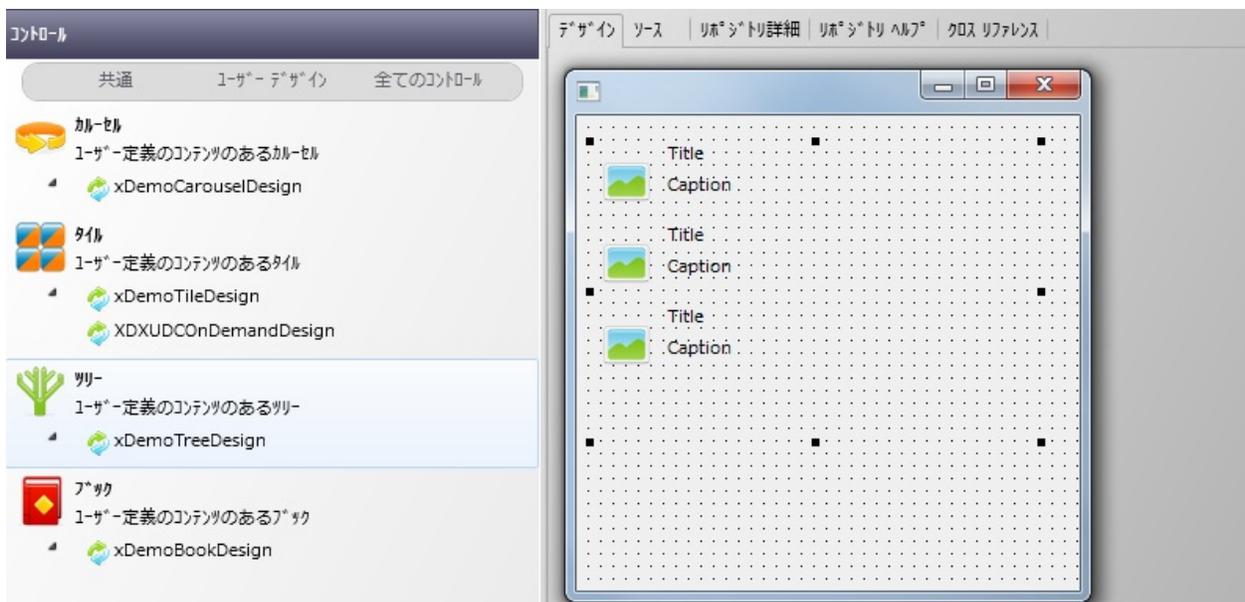


ユーザー定義のコントロール

ユーザー定義のコントロール (UDC) の作成が簡単になりました。

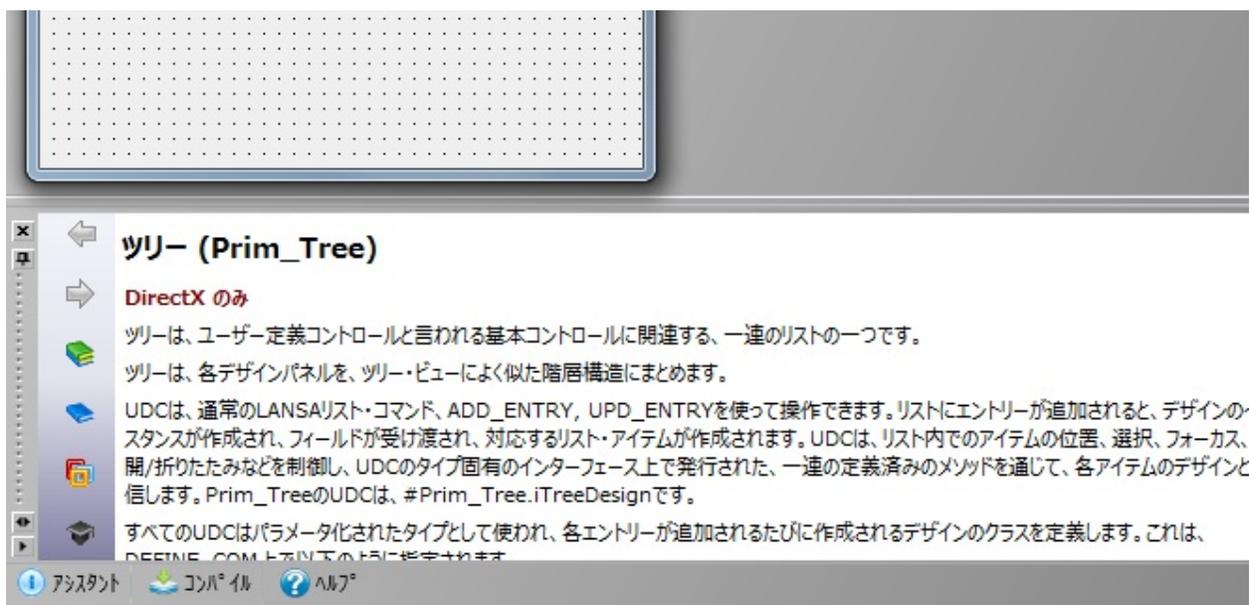


アプリケーションで使用できる、サンプルの UDC デザインを提供しています。



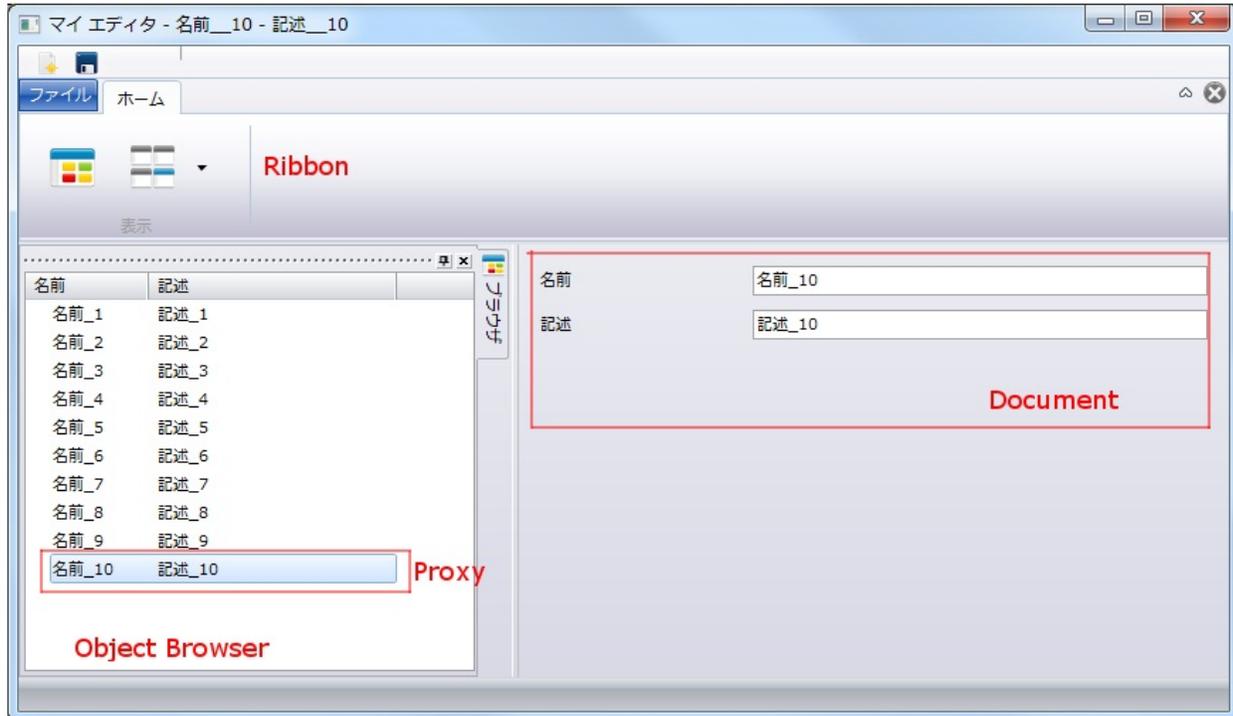
F1 機能ヘルプにより、今扱っているコントロールに関する情報を得る

ことができます。



CRUD ウィザード

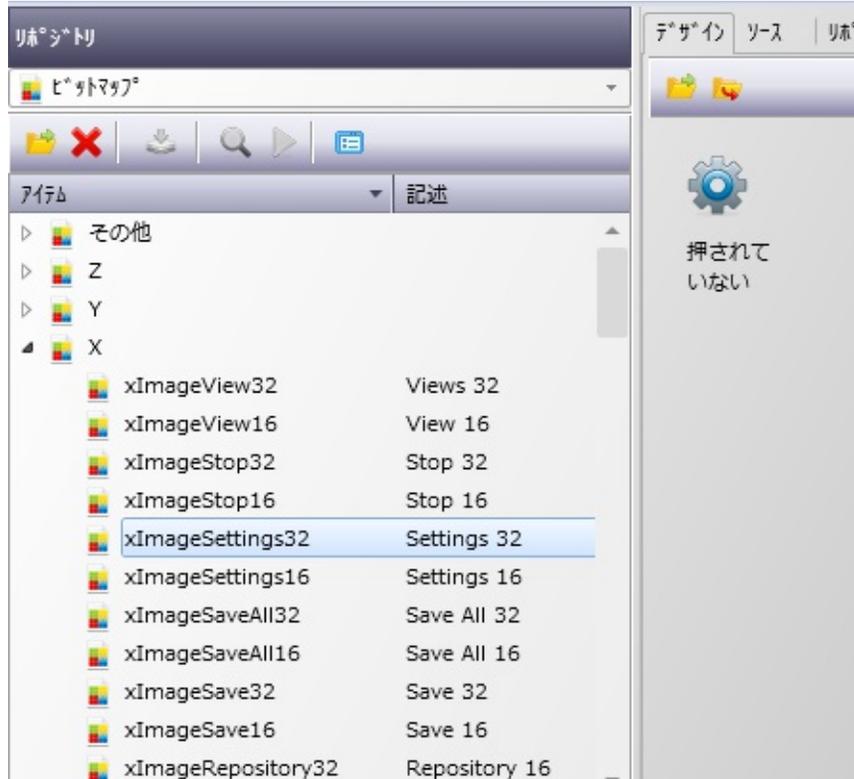
シンプルなウィザードを使用して LANSA IDE をモデルにした LANSA アプリケーションを作成することができます。



『Visual LANSA 開発者ガイド』の「ウィザードを使用したアプリケーションの作成」参照

新しいイメージとスタイル

新しいイメージがリッチクライアントアプリケーションで使用できるようになりました。



2つの新しいビジュアルスタイルを DirectXの機能を示す為に使用することができます。

XDemoStyles

スタイル フライアウト マウスオーバー 新規スタイル 新規レイアウト

ビジュアルスタイル

名前	記述
▶ J	
▶ K	
▶ L	
▶ M	
▶ N	
▶ O	
▶ P	
▶ Q	
▶ R	
▶ S	
▶ T	
▶ U	
▶ V	
▶ W	
▶ X	
▶ XDXStyles	DirectX スタイル
▶ xDemoStyles	Samples Styles
▶ Y	
▶ Z	

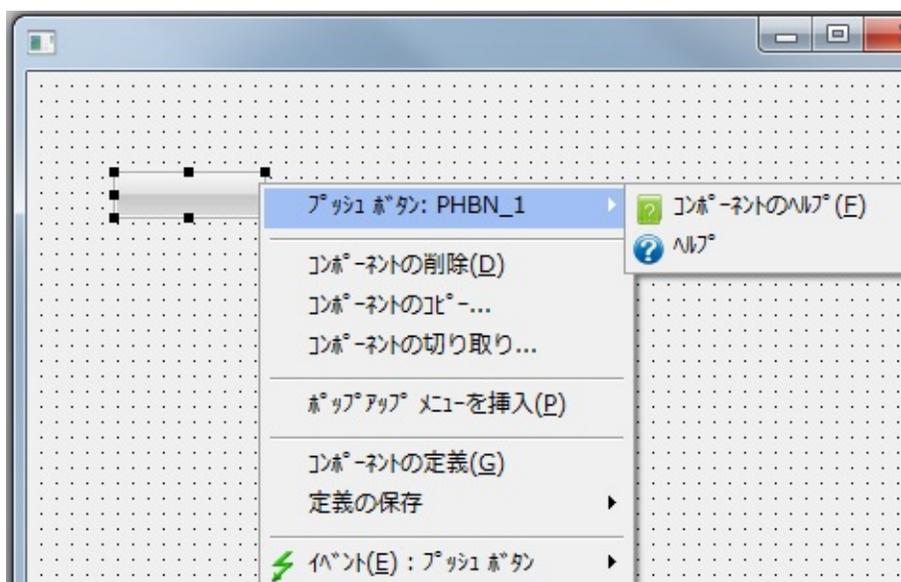
OK(O) キャンセル(C)

機能ヘルプの改善

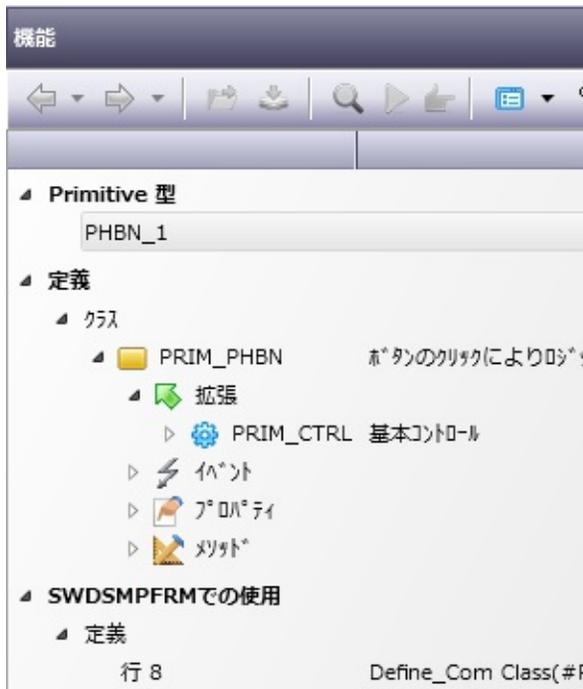
これまで、エディター・ウィンドウ下部のヘルプ・ペインに表示されていたコンテキスト依存ヘルプは、以下のファンクション・キーにより呼び出されておりました：F2で機能ヘルプ、F1で他の全てのヘルプの呼び出し。

機能とは、コンポーネントのプロパティ、イベント、メソッド、またはフィールドの長さやフィールド・タイプなどのことです。例えば、Caption プロパティに関する記述を表示させたいときには、そのプロパティにカーソルをあてて F2 を押しておりました。

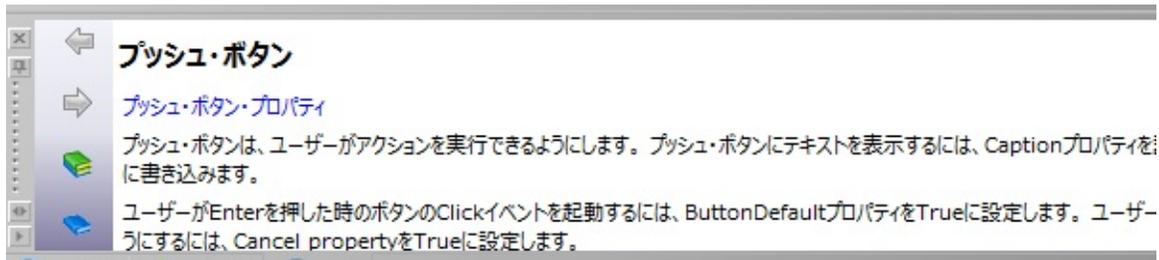
バージョン13 SP1からは、コンテキスト・メニューはフォーム上のコントロールを右クリックすると開くようになりました。



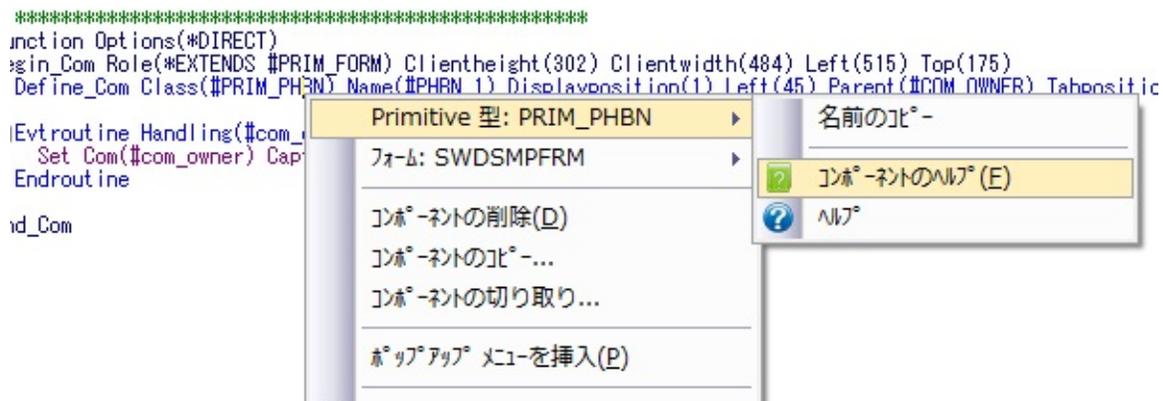
コンポーネントのヘルプを選択すると、機能ビューが表示されます。



ヘルプを選択すると、ヘルプビューが表示されます。



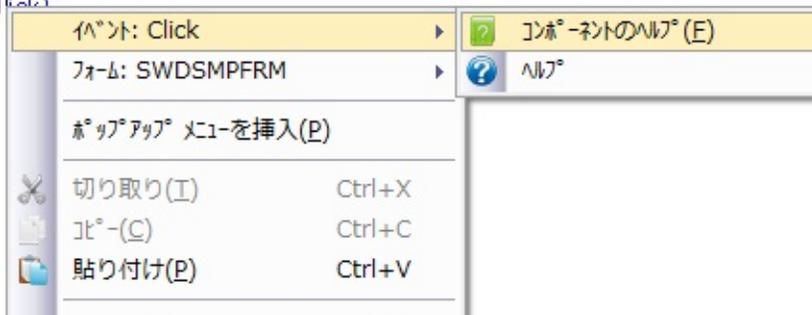
この機能は、ソースコードからも使用可能です。



または変数、イベント、プロパティ名からも使用可能です。

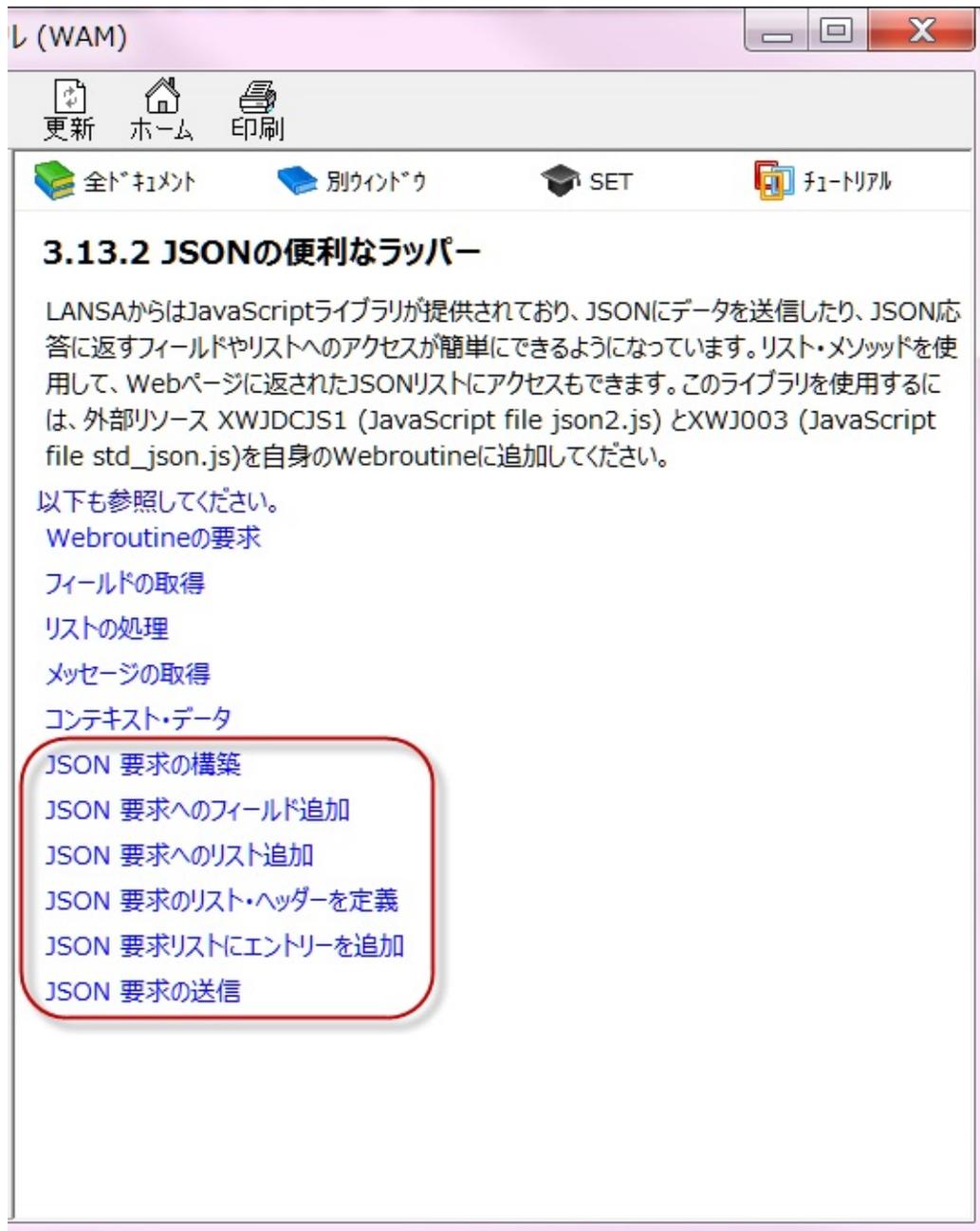
icon_owner) Caption(*component_desc)

Handling(#PHBN_1.Click)



JSON 形式での POST

ウェブラーチンは JSON 形式の POST をサポートするようになりました。



Web アプリケーション モジュール(WAM)

更新 ホーム 印刷

全トキメント 別ウインドウ SET チェートリアル

3.13.2 JSONの便利なラッパー

LANSAからはJavaScriptライブラリが提供されており、JSONにデータを送信したり、JSON応答に返すフィールドやリストへのアクセスが簡単にできるようになっています。リスト・メソッドを使用して、Webページに返されたJSONリストにアクセスもできます。このライブラリを使用するには、外部リソース XWJDCJS1 (JavaScript file json2.js) とXWJ003 (JavaScript file std_json.js)を自身のWebroutineに追加してください。

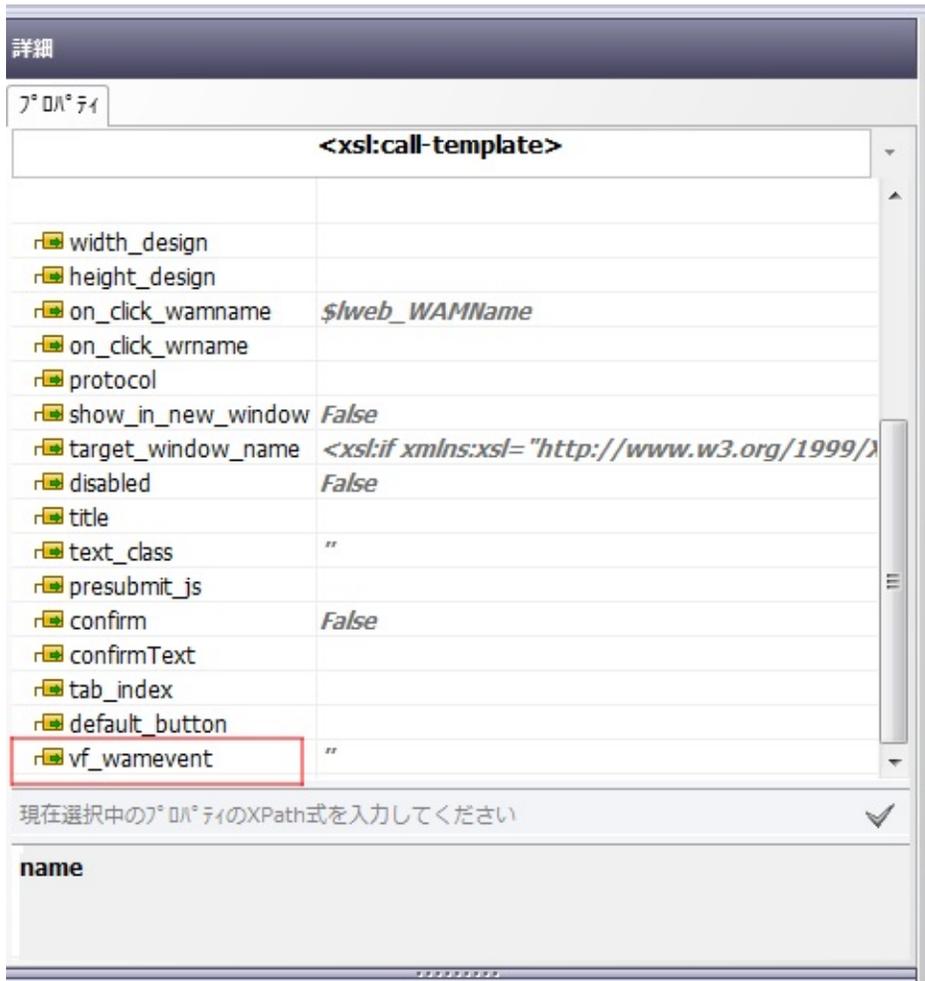
以下も参照してください。

- [Webroutineの要求](#)
- [フィールドの取得](#)
- [リストの処理](#)
- [メッセージの取得](#)
- [コンテキスト・データ](#)
- [JSON 要求の構築](#)
- [JSON 要求へのフィールド追加](#)
- [JSON 要求へのリスト追加](#)
- [JSON 要求のリスト・ヘッダーを定義](#)
- [JSON 要求リストにエントリーを追加](#)
- [JSON 要求の送信](#)

『Web アプリケーション モジュール(WAM)』ガイドの「[JSON サポート](#)」を参照してください。

vf_wamevent

リクエストを送るウェブレットは、VLF 開発者が WAM イベントをセットすることができる新しいプロパティ `vf_wamevent` を持つようになりました。



詳細

プロパティ

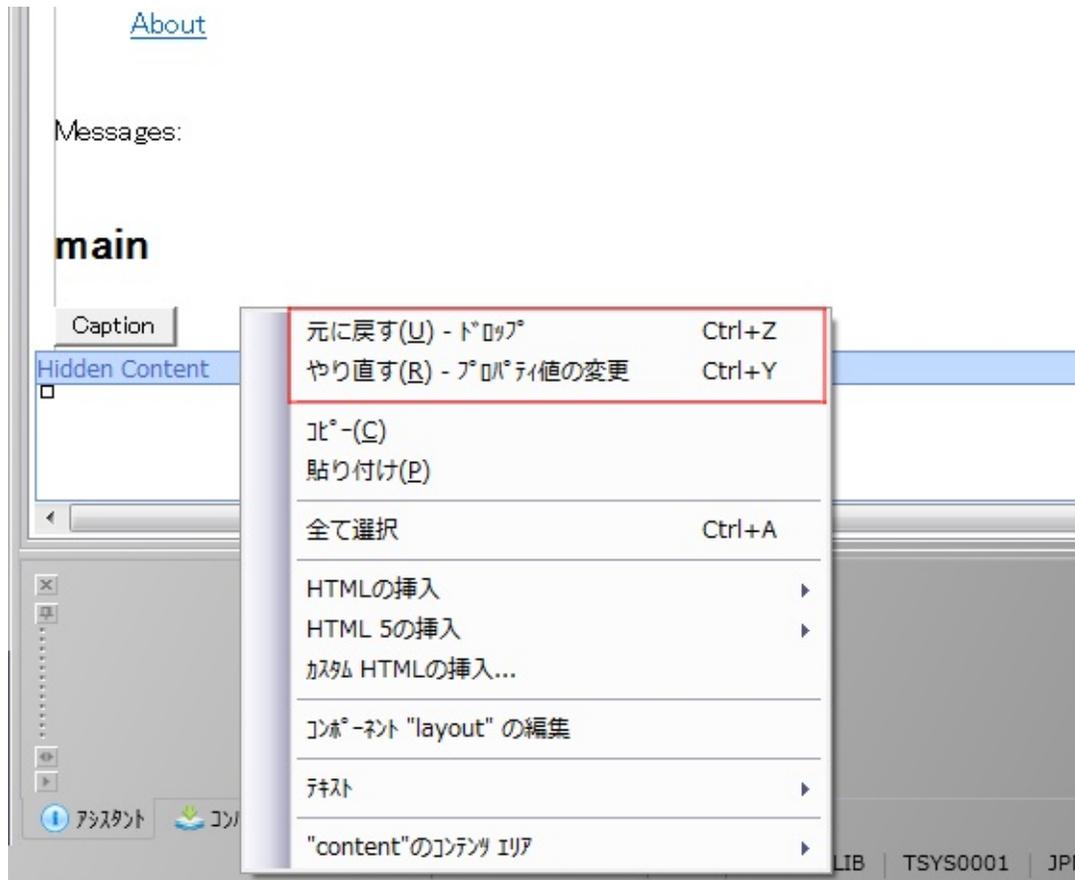
<xsl:call-template>	
width_design	
height_design	
on_click_wamname	<code><i>\$web_WAMName</i></code>
on_click_wname	
protocol	
show_in_new_window	<code><i>False</i></code>
target_window_name	<code><xsl:if xmlns:xsl="http://www.w3.org/1999/xsl"></code>
disabled	<code><i>False</i></code>
title	
text_class	<code><i>"</i></code>
presubmit_js	
confirm	<code><i>False</i></code>
confirmText	
tab_index	
default_button	
vf_wamevent	<code><i>"</i></code>

現在選択中のプロパティのXPath式を入力してください ✓

name

元に戻す/やり直し

WAM エディタの元に戻す/やり直し機能が改善されました。元に戻す/やり直しはコンテキストメニューで使用可能です。



リボンでも使用可能です。

履歴 ログ 実行 デバッグ 貼り付け 切り取り コピー 検索

ランタイム クリップボード

元に戻す (Ctrl+Z)
最後のアクションを元に戻す

Web Page XSL XML

[About](#)

Messages:

MSI 配布

V13 のアプリケーションを配布する前に、新しい MSI 配布メカニズムをしっかりと理解する時間をとってください。

バージョン番号は全てのDLLにコンパイルされる必要があります。コンパイルの設定ダイアログでバージョン番号を設定し、全てのアプリケーションを再生成する必要があります。全てのアプリケーションは一つの MSI に対して一つのパッケージ・バージョンでなくてはなりません。アプリケーションを部分ごとに異なるパッケージ・バージョンにしないようにしてください。



詳しくは、このマニュアルの「[配布](#)」及び、LANSAアプリケーション配布ツールガイドの「[配布機能の新機能](#)」を参照してください。

LANSA バージョン 13の新機能

ライセンス	新しいバージョン 13 のライセンスが、IBM i / Windows共に必要となります。
Microsoft DirectX ユーザーインターフェース	Microsoft DirectX ユーザーインターフェースが、Visual LANSАの高いエンドユーザー・エクスペリエンスとデータのよりリッチなビジュアルイゼーションを提供します。
.NET コンポーネントを使用する機能	LANSAのプログラムで .NET のユーザーインターフェースや非ユーザーインターフェース・コンポーネントを使用することができます。
モバイルのためのWAM	モバイルアプリケーションのためのWebアプリケーション・モジュール (WAM)を簡単に作成することができます。
UNICODEを使用した国際化	LANSA はフル Unicode サポートを提供します。
バージョン管理システムサポート	サードパーティのバージョン管理ツールへのLANSAのバージョン管理のインターフェースの第一段階が使用可能です。
配布	配布のために標準的なMSIパッケージとしてLANSAアプリケーションをパッケージ化することができます。
ロングネーム	ロングネームを使用することで、サードパーティとの統合や内容がわかるような名前をつけることができます。
RDMLX 機能拡張	新しい組み込み関数や基本コンポー

	ネット、言語のGET/SETコマンドなどが使用可能になりました。
Windows を中心とした開発	様々な機能拡張がLANSAになされ、それにより、更にWindows プラットフォーム上での開発が中心となります。
ファイルの機能拡張	IBM i 外部ファイルはUnicode、binary 及び varbinary フィールドをサポートするようになりました。
インポート・配布時のコレクション/ライブラリの変更	ファイルライブラリやコレクションはインポートや配布の際に上書きできるようになりました。
インストールと開発	新しいインストールと開発の機能が追加され、操作性と開発者の生産性を向上します。
Visual LANSA フレームワーク	新しいバージョンのフレームワークでは、DirectXのユーザーインターフェースを使用することができるようになりました。
LANSA Integrator	最新のLANSA Integrator がバージョン 13 と共に出荷されます。

エディション日付：2012年10月30日

© 2012 LANSA

ライセンス

バージョン 13 をインストールする場合、またはバージョン 13 にアップグレードする場合は、新しいバージョン 13 用のライセンスを取得する必要があります。

新しいライセンスは、IBM i / Windows 共に必要となります。サーバーライセンス、クライアントライセンス、ハードウェアキー（ダングル）が含まれます。

バージョン 13 へのアップグレードの前に、CPU 詳細を LANSА ライセンス担当に送り、新しいバージョン 13 のライセンスを取得してください。

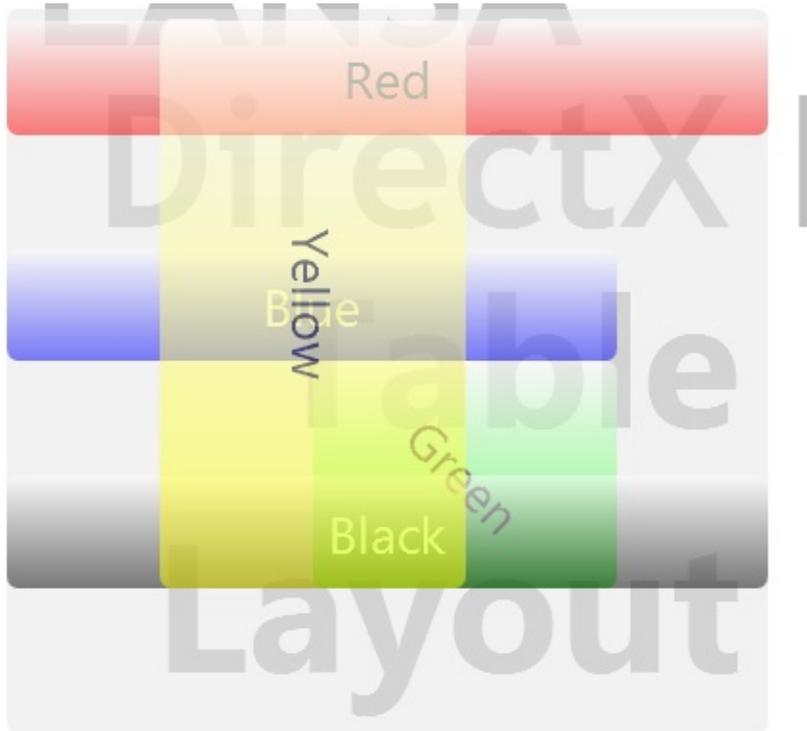
サーバーと Integrator のライセンスの適用方法が新しくなりました。

バージョン 13 では、全てのライセンスがひとつのライセンスの格納場所に統合され、管理がしやすくなります。同じ CPU 上の様々な LANSА 構成を統合したり、様々な CPU の複数ライセンスをひとつの場所に統合することができます。

詳しくは LANSА Web サイトの[製品ライセンスページ](#)(英語)をご確認ください。

Microsoft DirectX ユーザーインターフェース

Visual LANSA バージョン 13 では DirectX のレンダリング・エンジンを使用できます。DirectX は Windows に埋め込まれた API のコレクションであり、新しいデザインの可能性を開く、すばらしいグラフィック機能を提供します。



DirectX で何ができるかの例を見るには、[DirectX デモアプリケーション](#)を実行してください。

新しい Visual LANSA のユーザー定義コントロールを使い、開発者はユーザーインターフェースのほぼ完全な制御を行うことができます。例えば、

- 色のグラデーション、角のまるめ、透明性、不透明性などを使用できます。
- 動的なスタイルやマウスイベントにより、ユーザーインターフェースがマウス入力時、フーバー時、コントロールを離れた時にそれぞれアクションを起こすことができます。
- コンテキストメニューをただのメニューやアニメーションではなく、開発者がユーザーに対して価値のある視覚的なフィードバックを与えることができるものにすることができます。

[新しい Visual LANSA IDE](#)

動的スタイル

ブラシ

ユーザー定義のコントロール

テーブル・レイアウト

ポップアップ・パネル (Prim_PPNL)

スケーリング

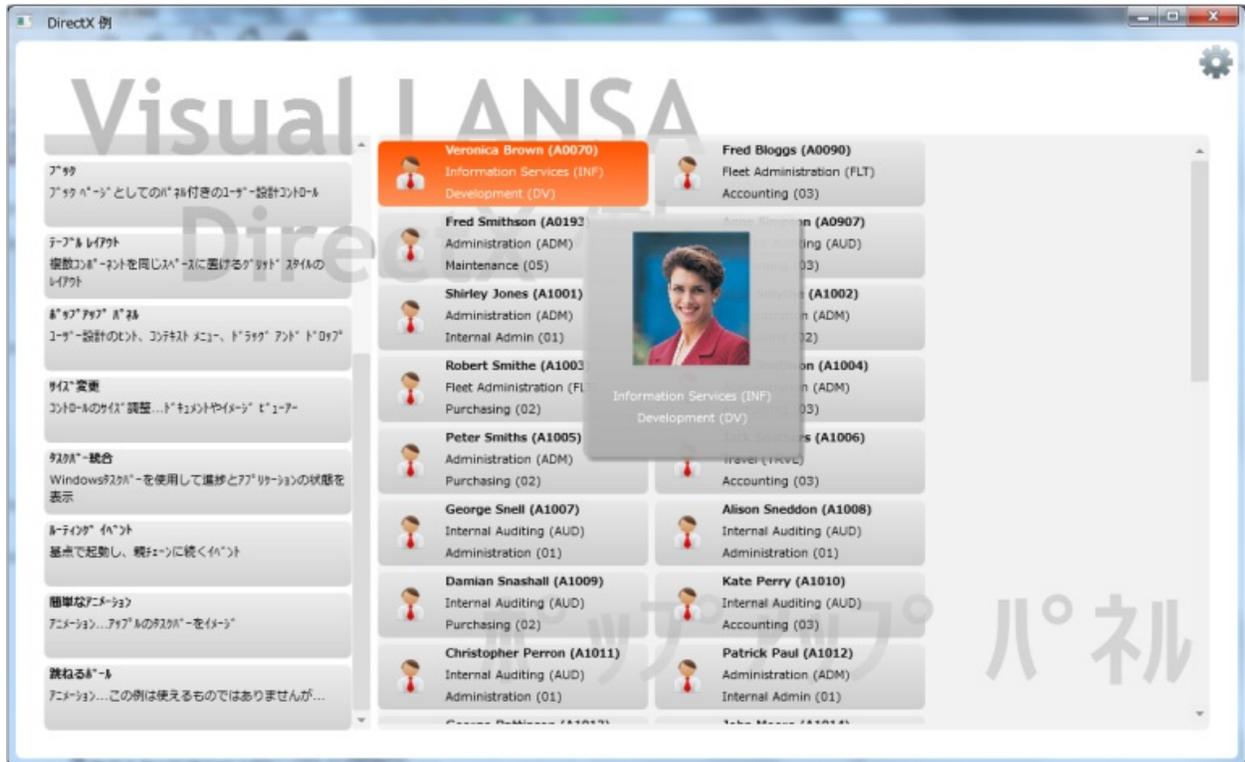
タスクバーの統合

アニメーション

DirectX の採用

DirectX デモアプリケーション

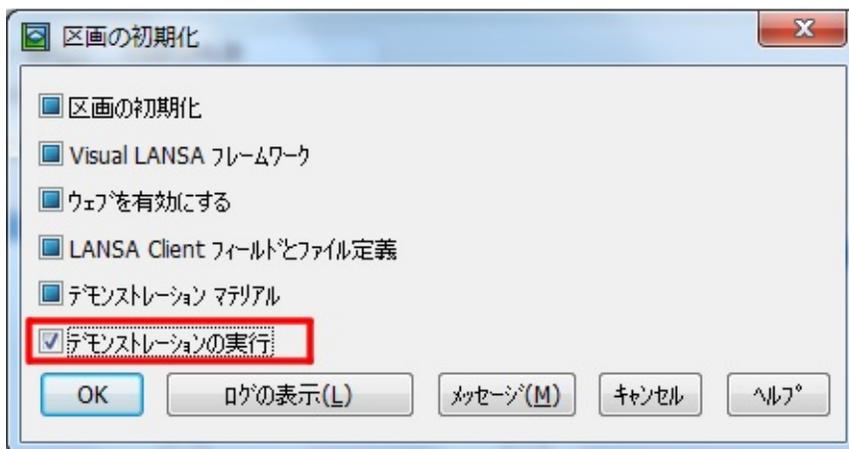
DirectX で何ができるかを、DirectX デモアプリケーションを実行して例を見てください。



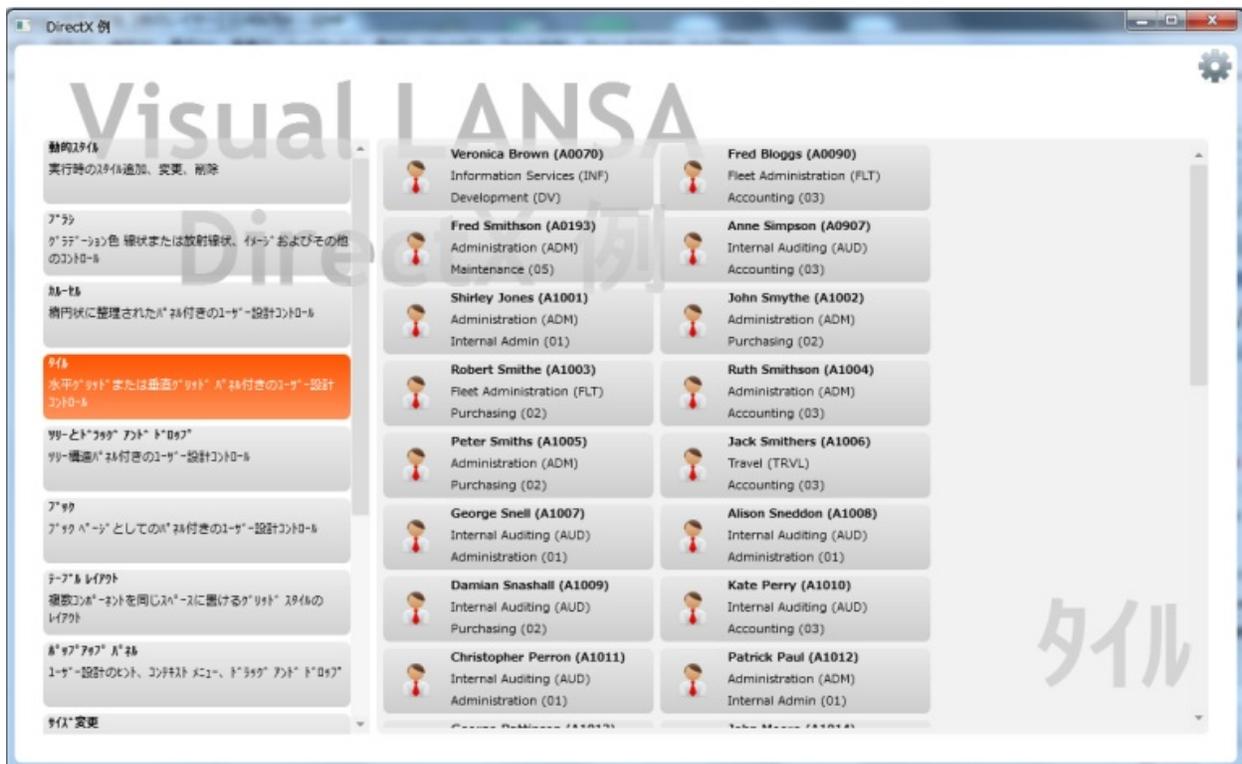
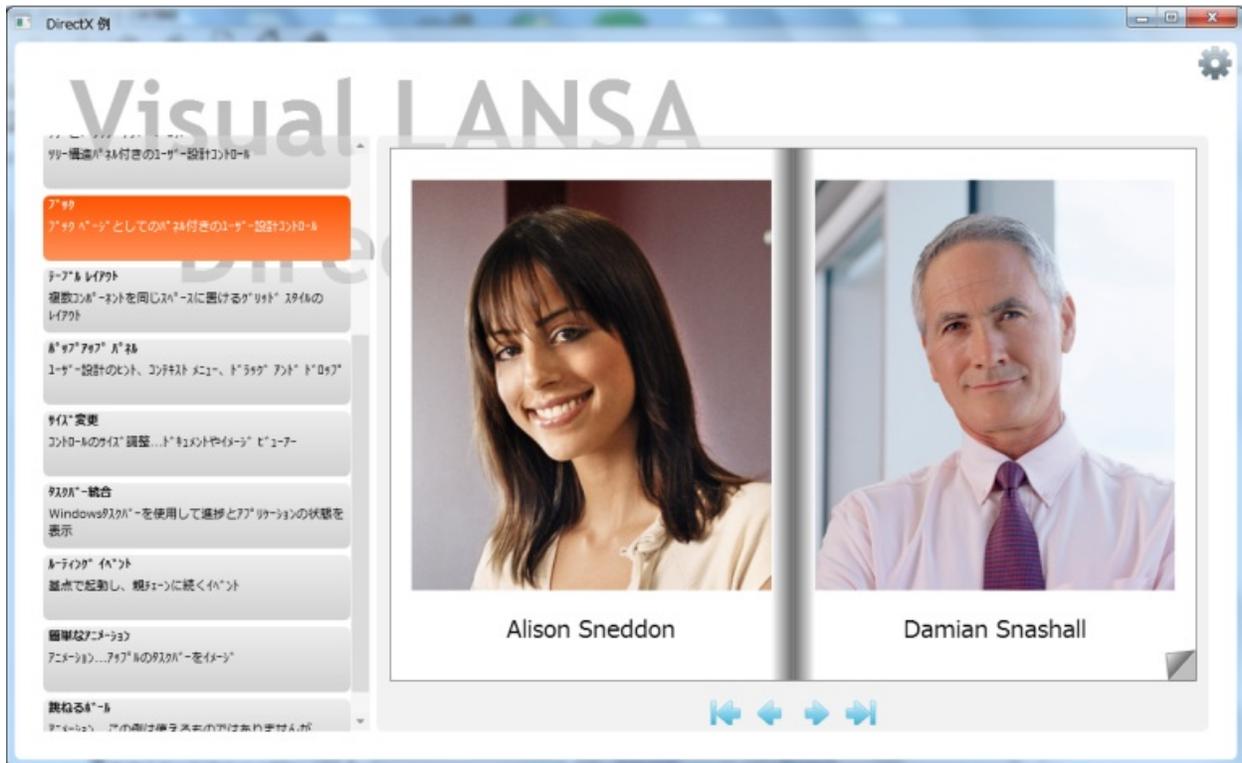
デモアプリケーションを実行するには、Windows上でLANSAにログオンする際に、[区画の初期化]を選択します。



[区画の初期化]ダイアログで、[デモンストレーションの実行]を選択します。



[OK]をクリックすると、[Visual LANSA DirectX 例]のアプリケーションが実行されます。





新しい Visual LANSA IDE

LANSA IDE (統合開発環境) は動的スタイル、アニメーション、レイアウト・ヘルパーや新しい DirectX ベースの機能をサポートしています。

リボン vs. メニュー & ツールバー

新しい IDE ではメニュー、ショートカット、ツールバーがリボンとしてひとつにまとめられます。

リボンはメニューやツールバーより少し多くの縦のスペースを使用しますが、この追加のスペースは、コンテキスト情報を表示できる、とても価値のあるスペースとなります。

例えば、スタイルヘルパーはフォーカス・コンポーネントがスタイルを使うコンポーネントの場合、スタイルがどのように見えるかを表示します。



リボンを使用すると、またバージョン12のレイアウトヘルパーとほぼ同様にスタイルを作成・構成することができます。しかし、リボンにヘルパーがあるので、シートを正面に持つてくることでこの情報を1クリックで見ることができます。

レイアウトヘルパーはまた、レイアウトヘルパーのダイアログを表示することなく、フォーカスコントロールの基本のレイアウト情報をリボンに表示します。



リボンはコンテキストが変更されると変更されます。例えば、ファイルが開かれた際には、ファイルは実行可能なオブジェクトではないため、

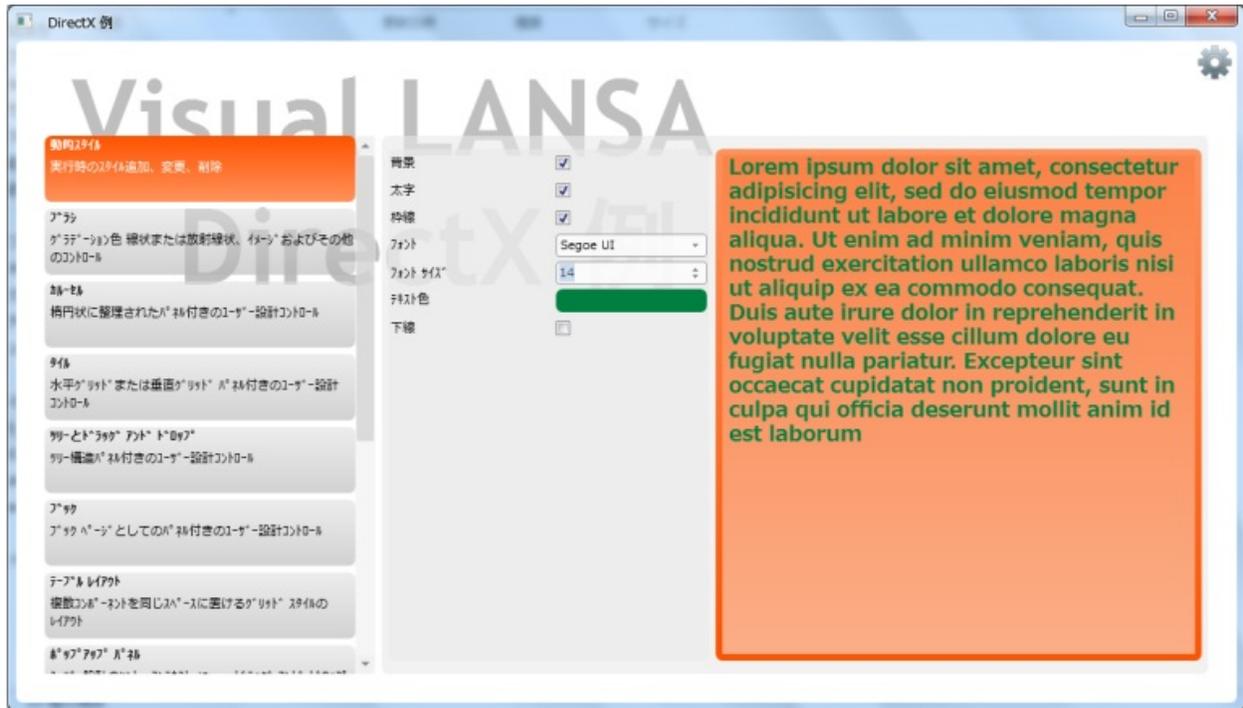
実行やデバッグのオプションは表示されません。

リボンはまた、全てのコマンドに簡単にアクセスすることができます。主な操作、保管、コンパイル、チェックイン、閉じるなどは、常に使用可能なクイックアクセス・ツールバーに表示されます。残りのものも、2〜3回のキー入力、もしくは2〜3回のクリックで、簡単に見つかります。

リボンで使用可能なプロパティや機能はよく使うものに限られています。他のプロパティを編集するための特別なビューが用意されています。

動的スタイル

コントロールに適用された際に既存のスタイル情報を単純に書き換えるビジュアルスタイルと違い、動的スタイルは実行時に追加したり削除したりすることができます。カスケード・スタイル・シートと同様です。



動的スタイルはビジュアルスタイルの一部として作成することができ、読み取り専用の機能としてアクセスすることができます。

```
Define_Com Class(#prim_vs.Style) Name(#LargeFonts) Bold(True) Fontsize(72)
```

```
#Panel.Style <= #MyStyles<LargeFonts>
```

もしくは、実行時に他のコンポーネント・インスタンスと同様に作成し、必要に応じてコントロールに適用することができます。このようにして作成されたスタイルは、完全に動的なものであり、ユーザーの好みに応じて実行時に修正することができます。

```
Define_Com Class(#prim_vs.Style) Name(#FontSize) Fontsize(10)
```

```
EvtRoutine Handling(#FontSize.Changed)
```

```
#DynamicStyle.FontSize := #FontSize
```

```
Endroutine
```

ひとつのコントロールに対して、複数のスタイルを追加したり削除したりすることができます。

```
Evtroutine Handling(#...)  
#Control.Styles.Add(#MyStyles<LargeFonts>)  
#Control.Styles.Add(#MyStyles<UnderlineItalic>)  
Endroutine
```

動的スタイルを使用して、角のまるめや罫線、水滴の影、前面ブラシ、背面ブラシ、不透明マスクなどの効果を定義することができます。スタイルは全てのコントロールの見た目を変更するのにアプリケーションレベルで適用することができます。

```
Define_Com Class(#prim_vs.Style) Name(#Label) Cornerbottomleft(3) Cornerbottomright(3)  
Cornertopleft(3) Cornertopright(3)  
Evtroutine Handling(#Com_owner.Createinstance)  
#Sys_Appln.Appearance.Label <= #Label  
Endroutine
```

MouseOverStyles

全てのスタイルには2セットのスタイルがあります。すなわち、Styles および MouseOverStyles です。

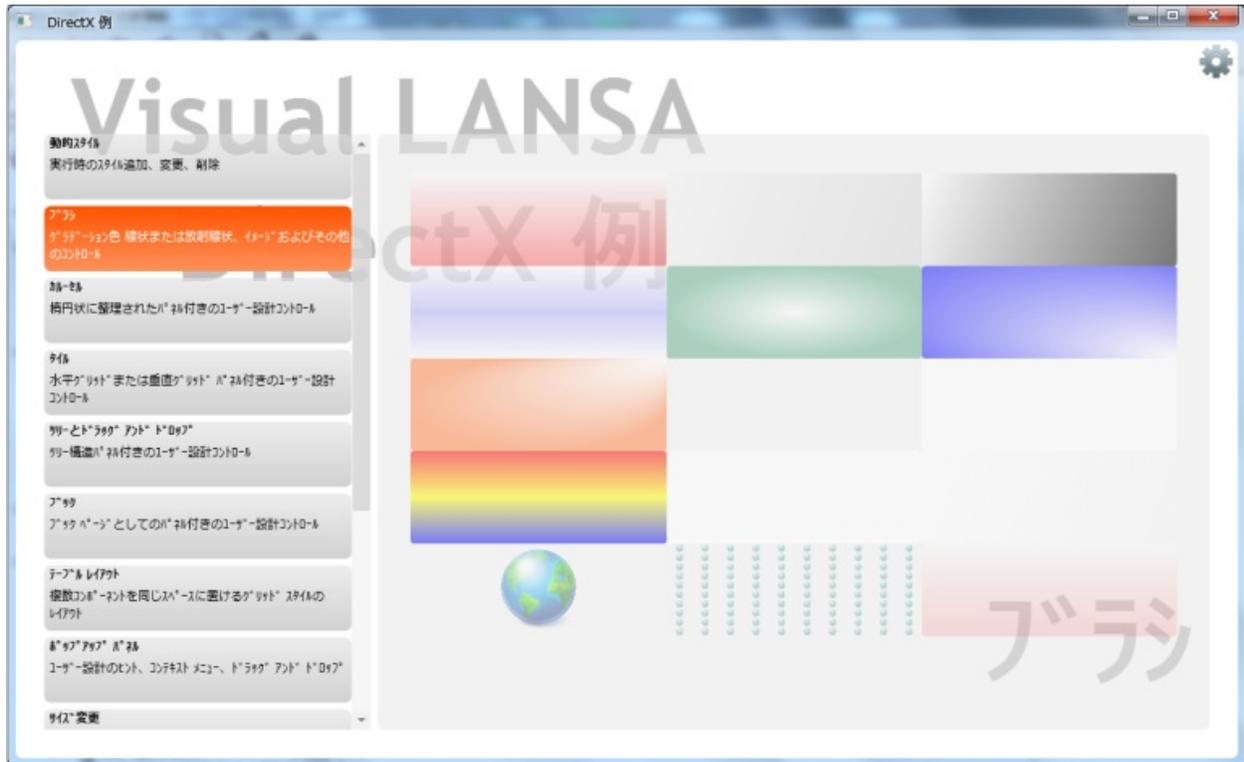
Styles はコントロールおよびその子コントロールに適用されます。

MouseOver Styles はマウスオーバー時、つまりマウスがコントロールの範囲に入ったときにコントロールに適用され、マウスリーブ時、つまりコントロールの範囲から離れた時に削除されます。これにより大量のマウス入力イベントを記述する必要なく、簡単な宣言コードを記述するだけでよくなります。

パネルやグループボックスのような複合コントロールには3つ目のスタイルのセット、PrivateStyles があります。Private Styles を使用すると、複合コントロールが、親から受け継いだスタイルを使用し続けたり、子コントロールには受け渡さない独自のスタイルを持つことができるようになります。例えば、コンテンツを赤にすることなく、グループボックスのキャプションを赤にすることができます。

ブラシ

ブラシを使用すると、色のグラデーション、画像、または他のコントロールを、背景、前景、スタイルの境界を埋めるために使用することができます。



使用可能なブラシには、いくつかのタイプがあります。

線状ブラシ

放射状のブラシ

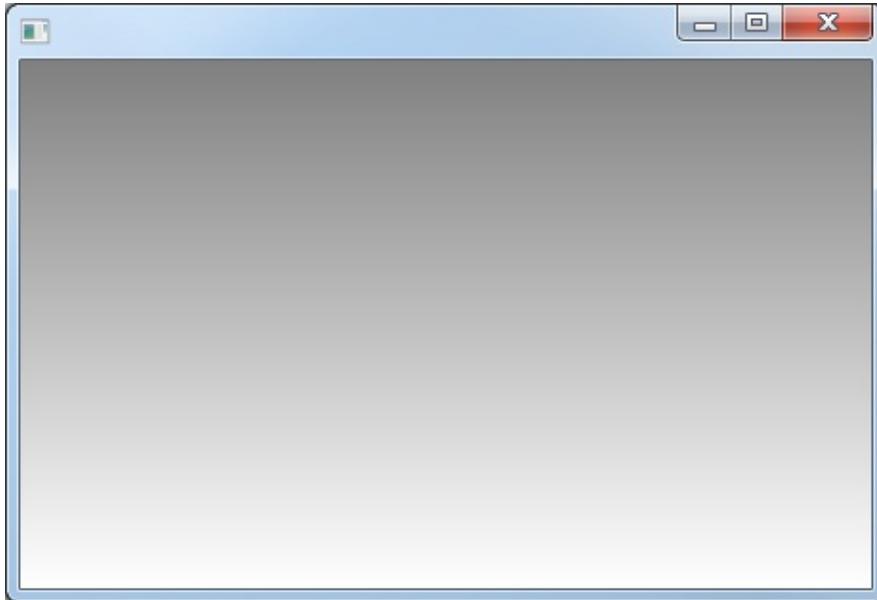
単色のブラシ

イメージブラシ

ビジュアルブラシ

線状ブラシ

下の画像は、灰色から白へと変わる線状ブラシを示しています。

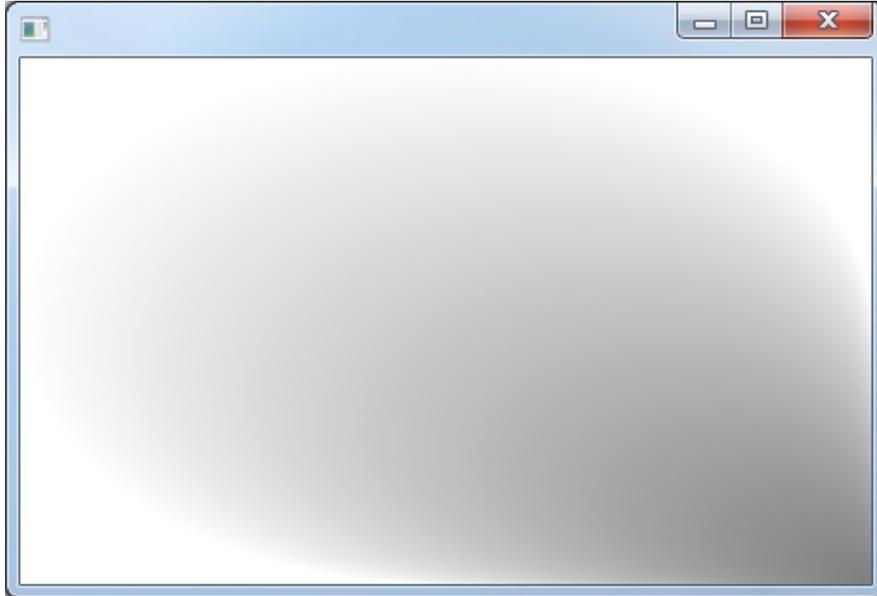


線状ブラシは開始と終了の座標により定義された線に従って色が変わっていくよう定義するのに使用されます。

```
Define_Com Class(#prim_vs.Style) Name(#Style) Backgroundbrush(#Brush)
Define_Com Class(#Prim_Vs.LinearBrush) Name(#Brush) Colors(#Colors)
Define_Com Class(#Prim_Vs.BrushColors) Name(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color1) Color(Gray) Parent(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color2) At(100) Color(White) Parent(#Colors)
```

放射状のブラシ

下の画像は、100,100（右下）を起点にグレーから白へと変わっていく放射線状のブラシを示しています。

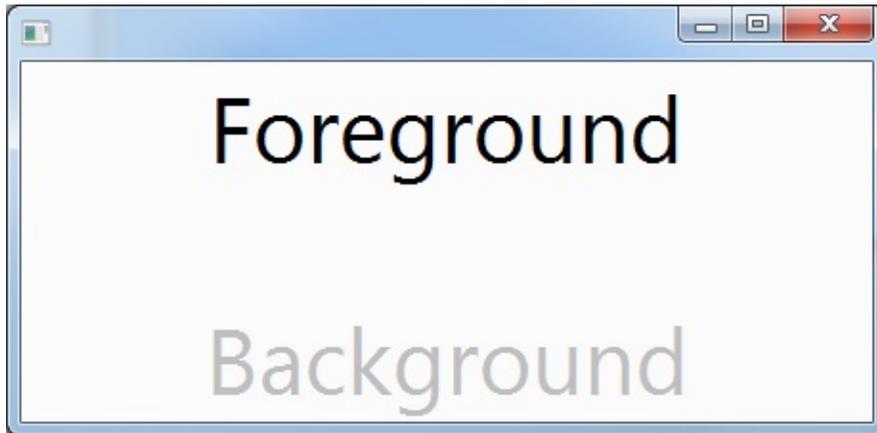


放射状のブラシは起点から広がる、開始と終了の座標により定義された線に従って色が変わっていくよう定義するのに使用されます。

```
Define_Com Class(#prim_vs.Style) Name(#Style) Backgroundbrush(#Brush)
Define_Com Class(#Prim_Vs.RadialBrush) Name(#Brush) Colors(#Colors) Originleft(100)
Originleft(100)
Define_Com Class(#Prim_Vs.BrushColors) Name(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color1) Color(Gray) Parent(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color2) At(100) Color(White) Parent(#Colors)
End_Com
```

単色のブラシ

下の画像は、不透明度75の単色の白色ブラシを示しています。ブラシは前景のテキストの上のパネルに適用されています。不透明度75の為、パネルを通して背景を確認することができます。

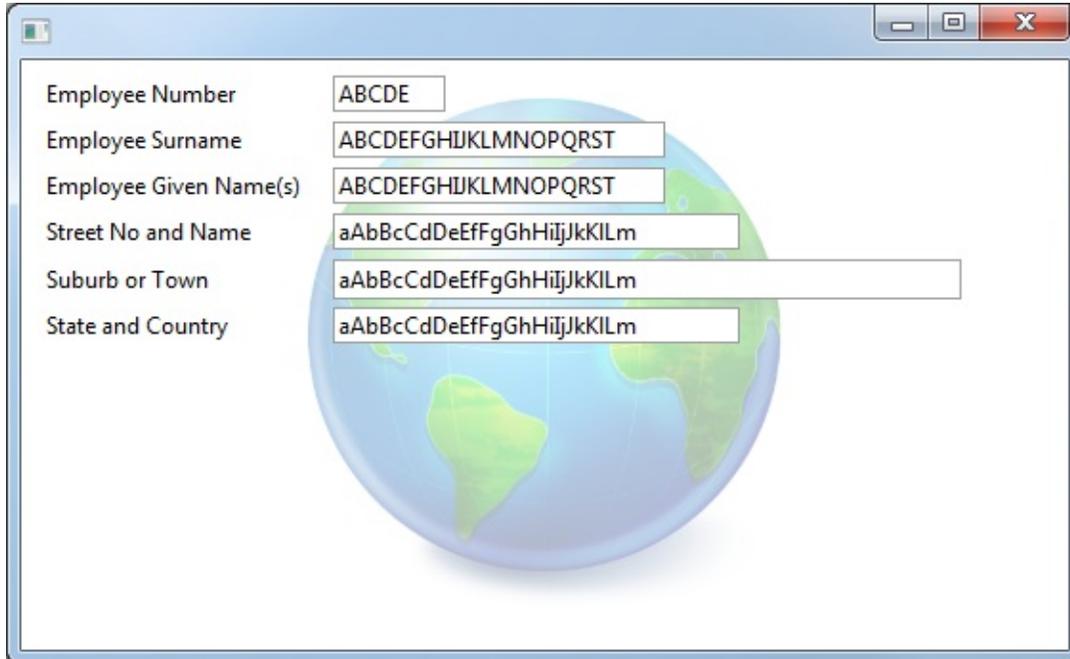


単色のブラシはひとつの色のブラシを定義するのに使用します。

```
Define_Com Class(#prim_vs.Style) Name(#Style) Backgroundbrush(#Brush)
Define_Com Class(#Prim_Vs.SolidBrush) Name(#Brush) Color(White) Opacity(75)
```

イメージブラシ

下の画像では、30%の不透明度のイメージブラシの上にフィールドを重ねています。



イメージブラシはブラシのコンテンツとして画像を描画するのに使用します。画像はサイズ変更したり、繰り返し使用したり、反射させたりすることができます。イメージブラシの典型的な使用方法は、透かしやフォームの背景などとして使用する方法です。

```
Define_Com Class(#prim_vs.Style) Name(#Style) Backgroundbrush(#Brush)
```

```
Define_Com Class(#Prim_Vs.imageBrush) Name(#Brush) Image(#Globe) Opacity(30) Sizing(BestFit)
```

ビジュアルブラシ

ビジュアルブラシを使用すると、他のコントロールのイメージをスタイルの背景もしくは前景として使用できるようになります。これは、画像のドラッグ&ドロップに特に便利です。イメージブラシと同様、コントロールの画像はサイズ変更したり繰り返し使用したりすることができます。



ユーザー定義のコントロール

ユーザー定義のコントロールはツリービューやグリッドに似たビジュアルリストの新しいフォームです。しかし、ユーザー定義のコントロールにはあらかじめ定義されている見た目がなく、ユーザーは構成する項目のデザインについて、自由な権限を持ちます。項目は通常のリストコマンド `Add_Entry`、`Dlt_Entry`、`SelectList` などを使用して追加します。

それぞれのユーザー定義のコントロールは独自のデザインのインターフェースを持ち(例. `#Prim_Tile.iTiledesign`, `#Prim_Tree.iTreedesign`)、デザインのインスタンスにフォーカスや選択、ツリーの展開・折りたたみなどに応答させるよう再利用可能パーツによって実装することができます。

使用したいリポジトリフィールドも定義することができます。それらは追加時にデザインのインスタンスにマップされます。

ユーザー定義のコントロールはデザインの1つのクラス、もしくは必要に応じてたくさんのクラスを使用することができます。

```
Define_Com Class(#prim_tile<#MyTileDesign>) Name(#Tile)...
```

タイル (Prim_Tile)

タイル項目はフローレイアウトのルールに従って制御されるグリッドパターンの中に並べられます。

 Veronica Brown (A0070) Information Services (INF) Development (DV)	 Fred Bloggs (A0090) Fleet Administration (FLT) Accounting (03)	 Fred Smithson (A0193) Administration (ADM) Maintenance (05)
 Anne Simpson (A0907) Internal Auditing (AUD) Accounting (03)	 Shirley Jones (A1001) Administration (ADM) Internal Admin (01)	 John Smythe (A1002) Administration (ADM) Purchasing (02)
 Robert Smithe (A1003) Fleet Administration (FLT) Purchasing (02)	 Ruth Smithson (A1004) Administration (ADM) Accounting (03)	 Peter Smiths (A1005) Administration (ADM) Purchasing (02)
 Jack Smithers (A1006) Travel (TRVL) Accounting (03)	 George Snell (A1007) Internal Auditing (AUD) Administration (01)	 Alison Sneddon (A1008) Internal Auditing (AUD) Administration (01)
 Damian Snashall (A1009) Internal Auditing (AUD) Purchasing (02)	 Kate Perry (A1010) Internal Auditing (AUD) Accounting (03)	 Christopher Perron (A1011) Internal Auditing (AUD) Administration (01)
 Patrick Paul (A1012) Administration (ADM) Internal Admin (01)	 George Pattinson (A1013) Administration (ADM) Internal Admin (01)	 John Moore (A1014) Administration (ADM) Purchasing (02)
 Bradley Woods (A1015) Administration (ADM) Internal Admin (01)	 Jack Turner (A1016) Fleet Administration (FLT) Administration (01)	 Gary Neave (A1017) Information Services (INF) Purchasing (02)
 Paul Zacharia (A1018) Group Accounts (GAC) Purchasing (02)	 Charles Dickens (A1019) Legal (LEG) Local Contracts (01)	 Adam Douglas (A1020) Administration (ADM) Internal Admin (01)
 Lisa McCully (A1021) Administration (ADM) Internal Admin (01)	 Kelly Thompson (A1022) Marketing (MKT) Purchasing (02)	 David Reid (A1023) Legal (LEG) Accounting (03)

ツリー (Prim_Tree)

ツリー項目はツリーに追加され、シンプルなリストのように振舞います。しかし、ParentItem を設定することで、項目を他のツリー項目の下にネストすることができます。項目が折りたたまれたり展開されたりする際に、ツリーは必要な項目を表示します。

The screenshot shows a web application interface with a search bar at the top. Below the search bar is a tree view of departments. The tree view includes:

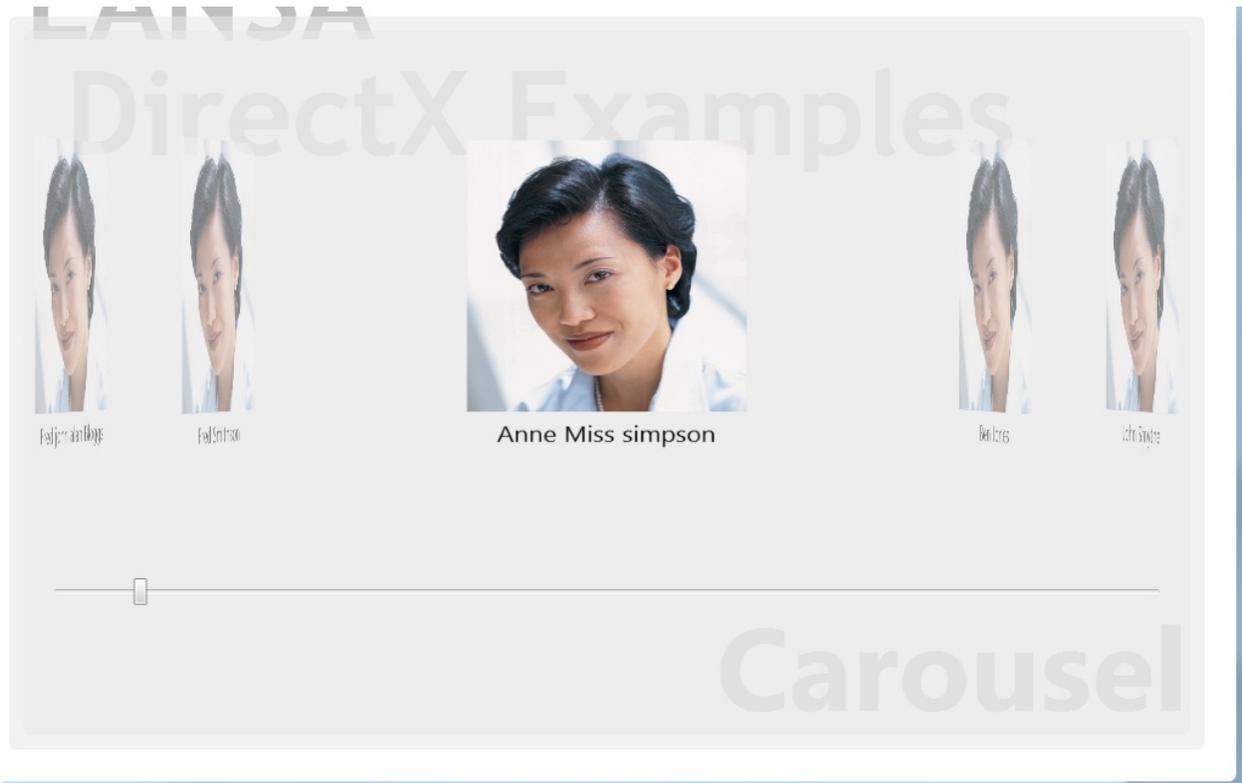
- Administration (ADM)
 - Internal Admin (01)
 - Purchasing (02)
- Accounting (03)
- Sales & Marketing (04)
- Maintenance (05)
- Personnel (06)
- Vehicle Maintenance (09)
- Internal Auditing (AUD)

Employee details are displayed for three employees:

John Smythe	20 Cobbitty Avenue, Werrington NSW 2100 Home Phone - 047 629 0442
Peter Smiths	72 Mullane Avenue, Baulkham Hills NSW 2147 Home Phone - 674 4316
John Moore	2 Burton Road, Lane Cove NSW 2100 Home Phone - 452 6392

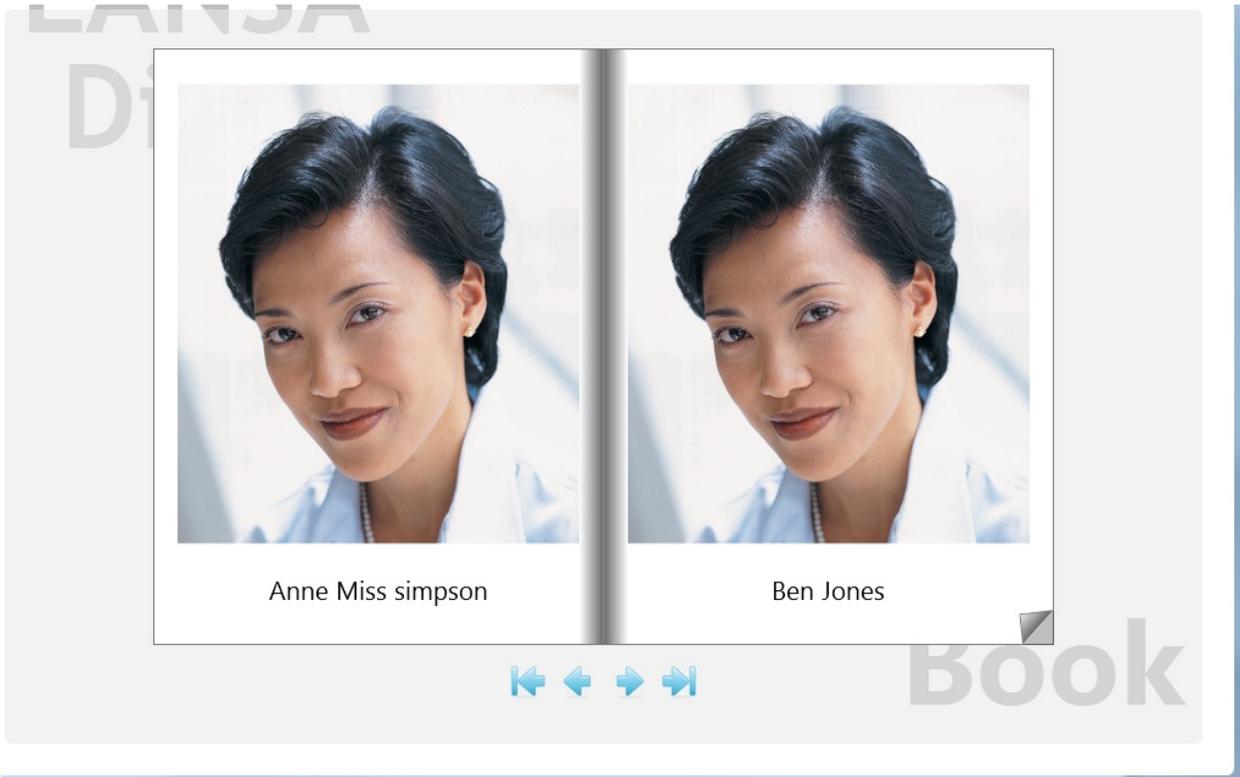
カルーセル

カルーセルは個別のデザインを、この場合は単純な画像とラベルを、線状または楕円状のパターンで表示します。簡潔にする為に、下の例では同じ画像を繰り返し使用しています。



ブック

ブックは個別のデザインを、この場合は単純な画像とラベルを、あたかも本のページのように表示します。簡潔にする為に、下の例では同じ画像を繰り返し使用しています。



テーブル・レイアウト

テーブル・レイアウトは新しいレイアウト・マネージャで、バージョン 12 のグリッドレイアウトや Microsoft Word のテーブルのコンセプトに似ています。

テーブルはいくつかの行と列に分割され、それぞれが使用可能なスペースのうち、あるパーセント、もしくは固定値のピクセルの幅を取ります。テーブルに含まれるコントロールにはそれぞれにレイアウト項目が与えられます。それらは列と行であり、行のスパンと列のスパンを指定することができます。

通常のレイアウトマネージャと異なり、テーブルはコントロールが同じスペースを使用することを許可しています。それにより、複雑な UI レイアウトの作成がとても簡単になります。

下のコードでは3列4行のテーブルを定義しており、4行目は高さを固定して定義されています。テーブル項目は #Control を管理しており、1行めに合うようにそれをサイズ変更します。

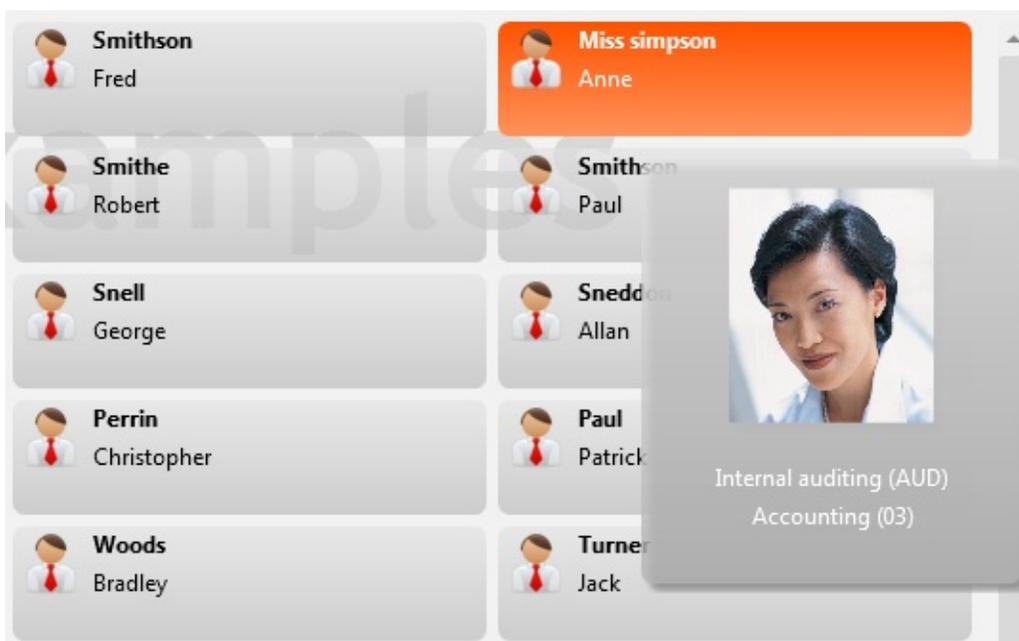
```
Define_Com Class(#prim_tblo) Name(#TableLayout)
Define_Com Class(#Prim_tblo.Column) Name(#TColumn1) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Column) Name(#TColumn2) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Column) Name(#TColumn3) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#TRow1) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#TRow2) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#TRow3) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#Trow4) Height(70) Parent(#TableLayout) Units(Pixels)
Define_Com Class(#Prim_tblo.Item) Name(#TableItem1) Column(#TColumn1) Columnspan(3)
Manage(#Control) Parent(#TableLayout) Row(#TRow1)
```

ポップアップ・パネル (Prim_PPNL)

ポップアップ・パネルはユーザー定義のコントロールのコンセプトを効果的に拡張したものです。あらかじめ定義されているタイプのヒントやポップアップメニュー、ドラッグイメージではなく、ポップアップ・パネルは代わりに完全にプログラミング可能な再利用可能パーツを表示するのに使用することができます。

全てのコントロールには Popup プロパティと HintPopup プロパティがあります。コントロールにポップアップをつけることで、右クリックすると、ポップアップが表示されます。同様に、ヒントのポップアップをつけると、ヒントが必要な際にポップアップが表示されます。

ポップアップ・メニューと同様に、ヒントが表示される直前に Prepare イベントが起動されます。それによりユーザーは必要に応じてポップアップのコンテンツを構成することができるようになります。



上の画像では、ポップアップ・パネルはアクティブな項目の追加情報を表示するのに使用されています。同様に下の画像では、右クリック時に、通常のコンテキストメニューだけでなく、内容の分かるツールバーも表示されています。



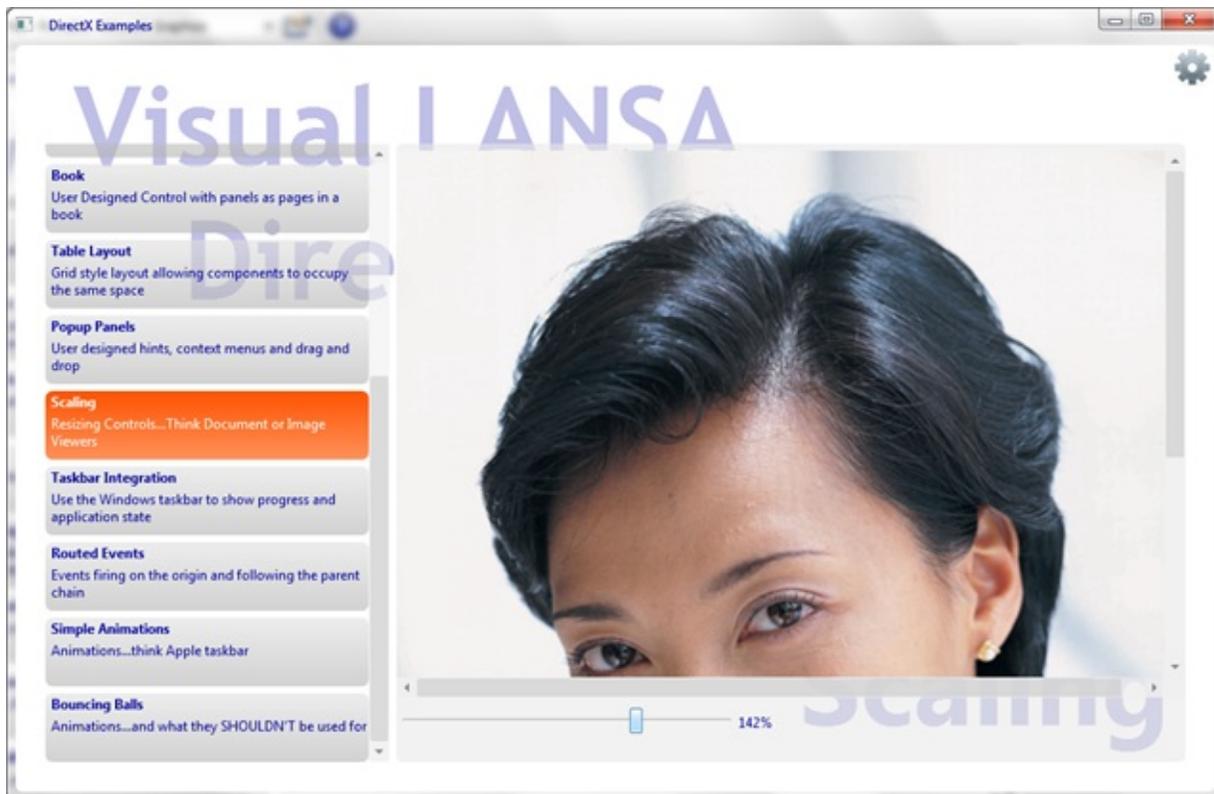
ヒントやポップアップ・メニューと違って、ポップアップは完全に機能する再利用可能パーツであり、ゆえにキーボード入力に対してフォーカスを取得したりアクションを起こしたりすることができます。これにより、Microsoft Officeで見られるようなコンテキスト処理が行えるようになります。

下のコードはヒントとコンテキストのポップアップがついたタイルを表しています。

```
Define_Com Class(#prim_Tile<#MyTileDesign>) Name(#Tile) Hintpopup(#HintPopup)
Parent(#COM_OWNER) Popup(#ContextPopup)
Define_Com Class(#prim_ppnl) Name(#HintPopup) Content(#HintPopupContent)
Define_Com Class(#PopupPanelEmployeeDetails) Name(#HintPopupContent)
Define_Com Class(#prim_ppnl) Name(#ContextPopup) Content(#ContextPopupContent)
Define_Com Class(#PopupPanelContextPopup) Name(#ContextPopupContent)
```

スケーリング

スケーリングにより、プロパティの高さや幅を変えることなく、コントロールを小さくもしくは大きく表示することができるようになります。これは通常、画像を見るときに便利ですが、全てのアプリケーションの機能の可視性を拡張するのに使用することもできます。コントロールはスケーリングされても、親コントロールのサイズに縛られます。



全てのコントロールには ScaleHeight、ScaleWidth、ScaleOriginTop、ScaleOriginLeft プロパティがあります。静的なプロパティとともに、コントロールには組み込まれたスケール・アニメーション・メソッドがあります。

下のコードでは、ボタンのスケールを50%拡大し、次に通常のサイズに戻し、ボタンが少しの間、画面外に飛び出る印象を与えます。

```
Define_Com Class(#prim_phbn) Name(#Button) Caption('Click Here') Displayposition(1)
Parent(#COM_OWNER) Tabposition(1)
Evroutine Handling(#Button.mouseEnter)
#Button.Scale( 150 150 150 )
#Button.Scale( 100 100 150 150 )
Endroutine
```

タスクバーの統合

Visual LANSA デスクトップ・アプリケーションは Windows のタスクバーと連携することができるようになりました。



Windows の TaskbarInfo を使用して、オーバーレイの画像と、進行の割合を指定することができます。ProgressStyle を使用して、進行の色と振る舞いも指定できます。

```
#Application.TaskBarInfo.ProgressStyle := Paused  
#Application.TaskBarInfo.OverlayImage <= #Pause16
```

上のコードではタスクバーを一時停止の状態に設定しており、そのため、下の画像のように、黄色で表示され、一時停止のイメージが重ねられています。



アニメーション

アニメーションは別段新しいものではありません。グリーンスクリーンでさえ、テキストが点滅します。

しかしながら、近代世界では、アニメーションはもう少し進化しており、画面の一定の部分に注意を引かなくてはならない場面で、エンドユーザーに対して機能をハイライト表示する手段とともに、画面に少し華やかさを加えます。

おそらくアニメーションの最も重要な機能は、スレッドの中で完全に動作するという点であり、アニメーションは他の長い処理が行なわれている間にも実行することができます。

アニメーションには、以下の2つのタイプがあります。

[画面切り替え](#)

[コントロールのアニメーション](#)

画面切り替え

画面切り替えはパネルを前面に持ってくるちょっと面白い方法であり、Microsoft PowerPoint のスライド画面切り替え方法に似ています。

従来では、前面にパネルを持ってきたい時には、以下のようなコードを記述していました。

```
#Panel2.Visible := True  
#Panel1.Visible := False
```

これは、2つ目のパネルを表示して現在表示されているものを隠します。処理はすぐに終わり、効果はシンプルです。

ただ、画面切り替えを使うと、他のパネルを表示させる間にパネルをフェードさせることができたり、画面の一番上から取り下げて他のを一番上に乗せたりすることができます。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Transition) Name(#Flip) Source(#Panel1) Target(#Panel2)  
Transitiontype(Flip) Parent(#Animation)  
Evroutine Handling(#Button.Click)  
#Animation.Start  
Endroutine
```

上のコードでは、フリップの画面切り替えを使った簡単なアニメーションを定義しています。これは、全てのコントロールの機能として使用可能であり、これにより、プログラムコードがより簡潔になります。

```
Evroutine Handling(#Button.Click)  
#Panel1.TransitionTo(#Panel2 Flip)  
Endroutine
```

コントロールのアニメーション

コントロールのアニメーションにより多様なアニメーション効果を得ることができます。ムーブ、フェード、回転、スケール、その他いろいろなタイプのアニメーションがあり、それぞれ、特定のコントロールに効果を与えるのに使用することができます。

しかしながら、重要なのは、アニメーションはコントロールをプロパティによって表現できない状態にはすることができないということです。ですので、コントロールのプロパティはアニメーションの間更新されず、アニメーションが完了したとき、コントロールは通常の Visual LANSARule に従います。

簡単な例として、レイアウトによって位置が決められるボタンはアニメーションによって好きな位置に動かすことができますが、アニメーションが終了したとき、レイアウトが制御を取り戻し、ボタンの位置を決定します。

簡単にする為に、いくつかのアニメーションはコントロールの機能として使用可能です。

```
Evtroutine Handling(#Button.Click)
#Button.Scale(200)
#Button.FadeOut
Endroutine
```

上のコードはボタンの幅を2倍にスケールしてから、フェードアウトさせます。

より複雑なアニメーションは複数のコントロールを動かす複合アニメーションを構築することで実現できます。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#Prim_anim.Opacity) Name(#ShowBanner) Duration(2000) Manage(#Banner)
Opacity(100) Parent(#Animation)
Define_Com Class(#Prim_anim.opacity) Name(#ShowGlobe) Duration(2000) Manage(#Globe)
Opacity(100) Parent(#Animation)
Define_Com Class(#Prim_anim.MoveFrom) Name(#BannerInFromLeft) Duration(2000)
Manage(#Banner) Parent(#Animation)
Define_Com Class(#Prim_anim.MoveFrom) Name(#GlobeInFromBelow) Duration(2000)
Manage(#Globe) Parent(#Animation)
```

この例は DirectX 例のスプラッシュ画面から取っています。バナーと地球を100%の不透明度にし、同時にそれらを画面外から画面内に動かすのに2秒かかります。

複雑なアニメーションは多くのコンポーネントから成っており、アニメーションが始まると全てが実行されます。各ピースには StartTime プロパティがあり、それを修正できるので、アニメーションを指定の順番で実行することができます。

DirectX の採用

バージョン 13 でアプリケーションを実行する際、バージョン 12 と全く同じように振舞い続けます。積極的に DirectX のレンダリング・オプションを使用することを選択する場合のみ、アプリケーションは変更されます。これは、アプリケーション、フォーム、またはパネルレベルで行うことができます。多くのユーザーにとっては、この切り替えはほとんどシームレスでしょう。しかしながら、DirectX のレンダリングが既存のアプリケーションの振る舞いに微妙に影響を与える場合もあるでしょう。

LANSA は「DirectX スイッチをフリックする」ということが可能な限りシンプルで変化のないものであるようにするために、また、バージョン 12 にユーザーインターフェースが近いままであるようにするために、苦労を惜しみませんでした。しかしながら、多くの新しい機能と、新しい基礎となる技術を採用することにより課せられる制約のため、いくつかの変更は避けられませんでした。

[DirectX レンダリングの採用](#)

[DirectX の変更](#)

[サンプルソース](#)

DirectX レンダリングの採用

Win32 の見た目を保持できるよう最大限の努力はなされたものの、状況により、Win32 の見た目を正確に反映した DirectX ランタイムを提供することは不可能ということがわかりました。

DirectX を使用可能にする前に、1つの質問に答える必要があります。「本当にそうする必要がありますか?」答えが「いいえ」であるなら、バージョン12 と同様の方法でバージョン13 を使い続けるべきです。

DirectX を使用可能にする

方針

テストを充分に実施する

DirectX を使用可能にする

Visual LANSA を使用して、DirectX をパネル、フォーム、またはアプリケーションレベルで適用することができます。DirectX が使用可能になると、Win32 コントロールに関係なく、全ての子パネルが DirectX を使用するようになります。

ランタイムを DirectX に設定すると、アプリケーション内のすべてのフォーム、パネル、コントロールは DirectX レンダリングを使用するようになります。

フォームが DirectX を使用するよう設定すると、子パネルとコントロールが DirectX レンダリングを使用するようになります。

パネルが DirectX を使用するよう設定すると、パネルとコントロールが DirectX レンダリングを使用するようになります。

方針

具体的に、DirectX の適用には、2つの方針があります。すなわち、全体的なもの、部分的なもの。

部分的なものは、元のもの比較的侵さない方針です。これにより、個別のフォームやパネルから、アプリケーションの残りの部分には影響を与えずに、新しいコントロールやアニメーションなど、DirectX 関連の機能を使い始めることができます。しかしながら、TrueType フォントの使用が強制されること、また、それにより、整合性が図れるようにアプリケーションの残りの部分のフォントを変える必要が発生する可能性があることは、特筆すべき点です。

全体的なアプローチはただランタイムが DirectX を使用するように切り替えられ、それにより、アプリケーション全体が DirectX を使用するようにレンダリングされるということを意味しています。

テストを充分に実施する

DirectX を開始する方針がどうであるかにかかわらず、アプリケーションの大部分はわずかに違った振る舞いをする可能性があります。フォントの変更などわずかなものもありますし、アプリケーションの一部が以前と同様に振舞わず、ランタイムエラーを起こす可能性もあります。

本番環境に移行する前に、アプリケーションの全体的なテストを実施することを強くお勧めします。

DirectX の変更

以下のセクションでは、個別の変更について詳細を述べると共に、それらの変更の理由とそれらが既存のアプリケーションにどのような影響を与えるかを説明しています。可能なところでは、それらの問題をどのように回避、対応するかについても説明しています。

[デフォルトの見た目](#)

[透明性と不透明性](#)

[ルートイベント](#)

[マウスイベント](#)

[ビジュアル・スタイル](#)

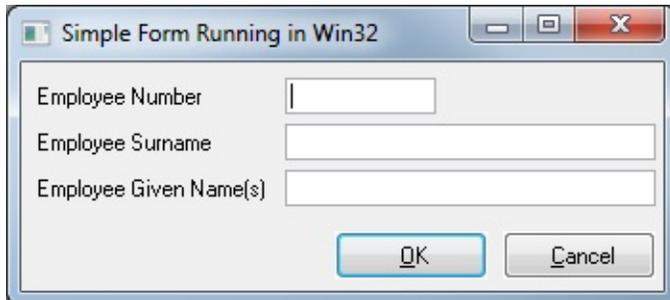
[True Type フォント](#)

[Win32 と DirectX \(ActiveX とグラフ\)](#)

[UpdateDisplay](#)

デフォルトの見た目

以下は同じフォームをバージョン13で実行した場合の画像です。一つ目は Win32 を使用しており、二つ目は DirectX を使用しています。



The screenshot shows a window titled "Simple Form Running in Win32". It contains three text input fields: "Employee Number", "Employee Surname", and "Employee Given Name(s)". Below the fields are two buttons: "OK" and "Cancel". The font used for the text is MS Sans Serif 9, which appears slightly pixelated and less smooth.



The screenshot shows a window titled "Simple Form Running in DirectX". It contains the same three text input fields and two buttons as the Win32 version. The font used for the text is Segoe UI 9, which appears much smoother and more modern.

このフォームのコードは、このドキュメントの「サンプルソース」のセクションで確認することができます。

機能的には、この2つのフォームは全く同じです。しかしながら、テキストに使用されるフォントは異なります。Win32 はデフォルトで MS Sans Serif 9 を使用するのに対し、Direct X はデフォルトで Segoe UI 9 を使用します。

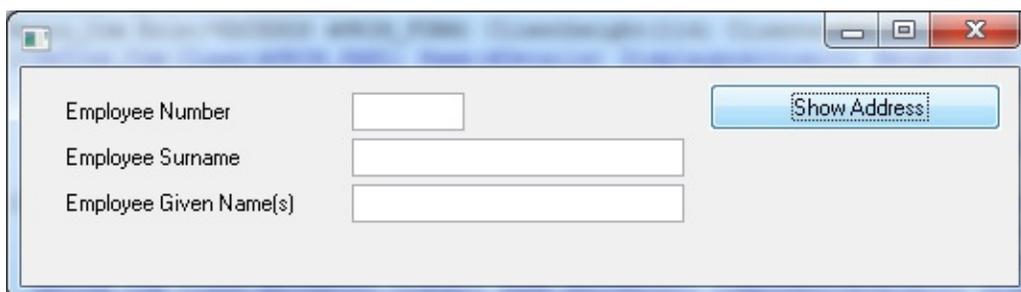
Ms Sans Serif は古いフォントで、昔、画面がより小さくより低い解像度であった時代に作成されました。結果として、最近の画面、最近の解像度で実行すると、近代的な True Type フォントの滑らかな縁や滑らかに丸まっている角に比べて、「ごつごつ」しているように見えます。

多くのユーザーにとっては、フォントの変更は全く重大なことではないかもしれませんが、Segoe UI がわずかに幅が広いので、画像から見て分かるように、テキストが折り返されたり、省略されたり、文字が切り捨てられたりする可能性があります。

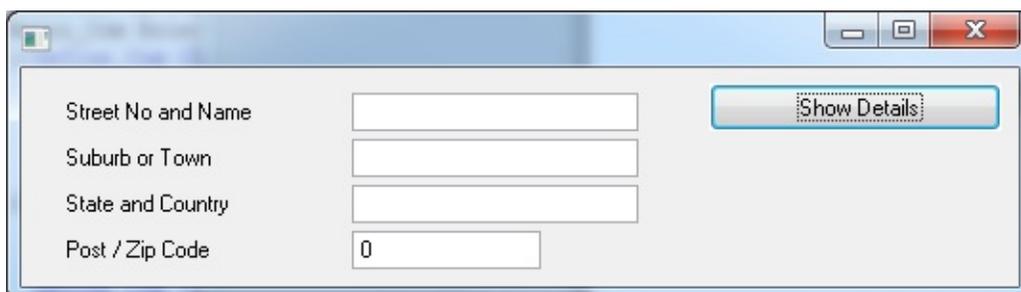
透明性と不透明性

DirectX レンダリングは透明性と不透明性を導入しました。省略値では、DirectX では特定のスタイルが適用されていない限り、全てのパネルやラベルが透明だと考えられます。この新しい見た目により、問題が発生する場合があります。

以下は、住所詳細と社員詳細の間を切り替えるシンプルなフォームです。ボタンをクリックすると、住所の詳細が使用可能になり、前面に表示されます。



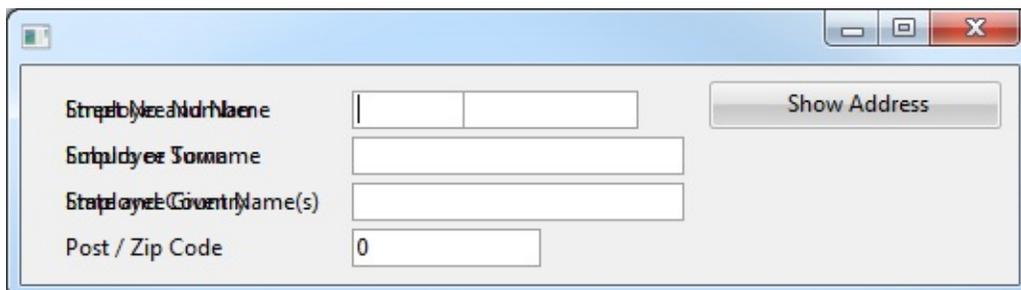
A screenshot of a Windows-style window titled "Employee Information". It contains three input fields: "Employee Number", "Employee Surname", and "Employee Given Name(s)". To the right of these fields is a blue button labeled "Show Address".



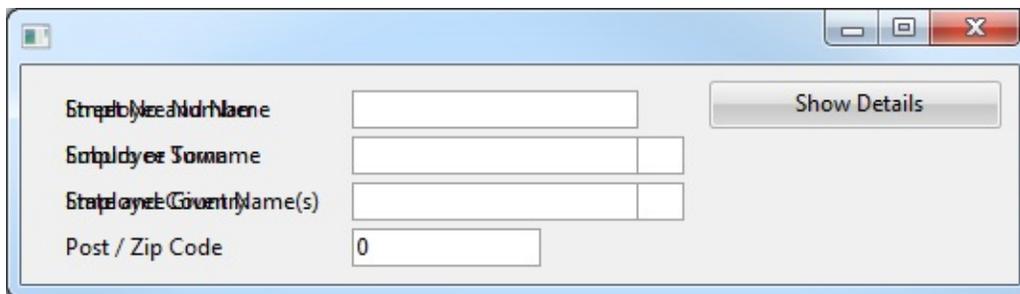
A screenshot of a Windows-style window titled "Address Information". It contains four input fields: "Street No and Name", "Suburb or Town", "State and Country", and "Post / Zip Code". To the right of these fields is a blue button labeled "Show Details".

このフォームのコードは、このドキュメントの「サンプルソース」のセクションで確認することができます。

しかし、DirectX レンダリングを使用した場合、結果は少々異なります。



A screenshot of a Windows-style window titled "Employee Information". It contains three input fields: "Employee Number", "Employee Surname", and "Employee Given Name(s)". To the right of these fields is a grey button labeled "Show Address". The text labels for the input fields are partially obscured by overlapping text, likely due to rendering issues with transparency.



The image shows a screenshot of a Windows-style dialog box. It has a title bar with standard minimize, maximize, and close buttons. The main area contains four input fields for address information: 'Street Name', 'City Name', 'State/Country Name(s)', and 'Post / Zip Code'. The 'Post / Zip Code' field contains the number '0'. To the right of these fields is a button labeled 'Show Details'.

住所のパネルと詳細のパネルの `DisplayPosition` に関係なく、両方とも表示されています。

この省略値は複雑なレイヤー構造のフォームを構築し、その上で透かし画像やそれに適用された背景を見えるようにしたいということから始まっています。それらが不透明だった場合には、全てのパネルとラベルを確認して、透明性のスタイルを適用しなくてはならなくなるからです。

この状況の簡単な回避方法は、アクティブでないパネルを `Enabled(False)` ではなく、`Visible(False)` に設定することです。

ルートイベント

複雑な再利用可能パーツや、特に新しいユーザー定義コントロールのデザイン画面のコーディングを簡潔化する為に、コントロール上で検出されたイベントは親チェーンに渡されるようになりました。

ユーザー定義コントロールではよく、表示されるパネルはラベルや画像などから構成されています。しかし、既存のイベント処理では、各ラベルがクリックイベントをもち、受け渡しはしませんでした。それにより、ユーザーは、全ての子コントロールについて、全てのクリックイベントを記述する必要がありました。イベントを親チェーンに受け渡すことで、コーディングはとても簡潔化されます。

もちろん、どのコントロールがイベントを最初に起動するかを知っておく必要があります。EVTROUTINE コマンドは既に Com_Sender セレクタを持っていますが、これは、イベントを起動したコントロールについてしか報告しません。そのため、Origin セレクタが追加されました。どれだけたくさんの親レイヤーが使用されていたとしても、Origin はイベントが実際に開始されたインスタンスへの参照を持ちます。

```
EvtRoutine Handling(#Com_owner.Click) Origin(#Origin)
```

```
...
```

```
Endroutine
```

このルールの唯一の例外は、イベントが再利用可能パーツの外に出た場合です。再利用可能パーツはそれ自体はブラックボックスで、カプセル化のルールにより、再利用可能パーツ内で起こることは、再利用可能パーツ内にとどまらなくてはなりません。そのため、この場合には、親についていえば、起源は再利用可能パーツ自体となります。

このドキュメントのリリース時点では、コントロールは再利用可能パーツをまたがっても使用することができます。これは既知の問題です。

明らかなのは、このイベントの振る舞いの変更により、副次的な効果があるということです。子と親コンポーネントの両方にクリックイベントがある簡単な例では、Win32 では2つのイベントは別々のもののままになります。しかし、DirectX の処理とイベントルートでは、子をクリックすると、子と親の両方のクリックが起動されることになります。これは通常の状態ではなく、お客様が直面する可能性は低いですが、そうはいつでも、考えられる状況であり、対応する必要があります。

望まないイベントの広がりに対抗する為、EVTROUTINE に Handled Selector が追加されました。Handled に true と設定することで、イベント

は処理されているルーチンを超えては受け渡されなくなります。

```
EvtRoutine Handling(#Button.Click) Handled(#Handled)
```

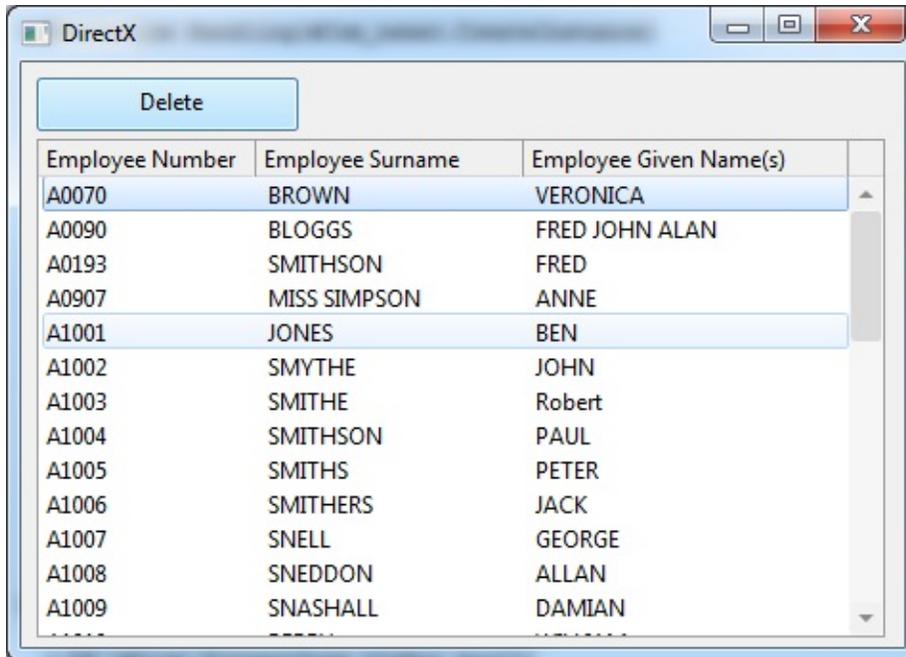
```
* Stop the event going any further up the parent chain.
```

```
#Handled := True
```

```
EndRoutine
```

マウスイベント

アプリケーションが DirectX を使用している場合、リストコントロール (ツリー、グリッド等) の表示方法に変更があります。中でも注目すべきは、マウスがリスト内の項目の上にある場合、項目がハイライト表示されることです。下の図では、リストの最初の項目が、FocusItem です。5 番目の項目上にマウスがあり、それが CurrentItem となっています。



このフォームのコードは、このドキュメントの「サンプルソース」のセクションで確認することができます。

全てのLANSAのリストはリストから対応するフィールド値をマップし、対応する変数へマップする CurrentItem の概念を使用しています。CurrentItem は事実上、リストの中で最後に処理した項目を表しています。これは、クリックされた最後の項目であるか、FocusItem である可能性もありますし、あるいは、Selectlist ループの中で処理された最後の項目かもしれません。

上記と同様な処理は、以下のようなコードで削除ボタンが現在の選択された項目を削除する場合の処理によくあります。

```
Evtroutine Handling(#Delete.Click)
If (#List.CurrentItem *IsNot *null)
Dlt_Entry Number(#List.CurrentItem.Entry) From_List(#List)
Endif
Endroutine
```

このコードは実際は CurrentItem 及び FocusItem は全く同一のものである

ことを示しており、DirectX 以前のほとんどのシナリオに当てはまりません。

しかし、DirectX では、マウスオーバー処理は全てのリストに追加され、すぐに使えるようになります。関連するイベントルーティンが記述されていなくても、ランタイムは何が CurrentItem であるのかを決定し、それが変数の値に反映されます。

Selectlist の外側では、データが破損している可能性があるため、CurrentItem に頼ることはいい方法ではありません。ひとつの選択リストには、常に FocusItem を使用してください。

ビジュアル・スタイル

ビジュアル・スタイルはこれまで、はっきりしない理由のために、BorderColor プロパティを無視してきました。テーマが適用されない限り、境界線は黒く塗られていました。テーマが適用された場合には、テーマが変更され、そのテーマの境界線の色が適用されました。

DirectX では、指定した境界線の色が適用されるようになりました。これにより、ビジュアル・スタイルに誤った値を指定しているお客様では、問題が発生する可能性があります。これを回避する明確な方法は、ビジュアル・スタイルを修正することです。

このドキュメントをリリースする時点では、ビジュアルスタイルのBorderStyle は DirectX ランタイムで無視されます。全ての境界線が単一線で表示されます。

True Type フォント

DirectX レンダリングは True Type フォントのみをサポートします。これは単に、基礎となる Microsoft のテクノロジーがそうであるからです。True Type 及び Open Type、True Type 拡張は、業界の標準であり、使用されるフォントサイズにかかわらず、滑らかにレンダリングするように設計されています。

フォントがレンダリングできないところでは、Visual LANSА は Segoe UI を使用します。

True Type ではない MS Sans Serif などのフォントには、代わりとなる近代的な True Type が存在するはずですが、MS Sans Serif の代わりとなるものは、Microsoft Sans Serif です。

DirectX を採用したい場合には、アプリケーションを True Type フォントを使用するように変更することを強くお勧めします。これにより、使用可能なスペースにテキストが入りきらないなどの問題が発生する可能性があります。行なった変更を確認することをお勧めします。

Win32 と DirectX (ActiveX とグラフ)

同じ UI スペースで Win32 と DirectX の両方を操作しようとした場合、互換性の問題が発生します。フォントに関しては、これは基礎となるテクノロジーの違いによるものです。

多くのお客様では、DirectX は既存のアプリケーション上の個別のフォームや、または既存のフォームに埋め込まれているパネルに採用されているでしょう。DirectX は Win32 の中で正しく動く為、このテクニックは問題ありません。しかし、逆の場合は、そんなに簡単ではありません。Win32 コントロールが、思ったようには振舞わないからです。

Win32 コントロールは DirectX と同じレンダレベル部分を占めることができず、そのため、異なったレベルに配置されます。これにより、子の Win32 コントロールが親よりも大きい場合、スクロールの問題が発生します。下の画像では、ブラウザはスクロールされたパネルの親となっています。ブラウザの上部が右の画面スクロールバーの上部と一致していることを確認してください。これは Win32 アプリケーションであるため、ブラウザは正しく留められています。



このフォームのコードは、このドキュメントの「サンプルソース」のセクションで確認することができます。

しかし、DirectX で 同じフォームを実行した場合(下図参照)、パネルを

スクロールすると、ActiveX が動き、留まりません。ブラウザが右側のパネルのスクロールバーより上にあることを確認してください。



この問題を実際に解決する唯一の方法は、おそらくレイアウトマネージャを使用して、Win32 コントロールを適切なサイズに変更し、親のサイズを超えないようにすることです。

このドキュメントのリリース時点では、メモ (Prim_memo) とリストビュー (Prim_ltvw) はまだ Win32 コントロールです。

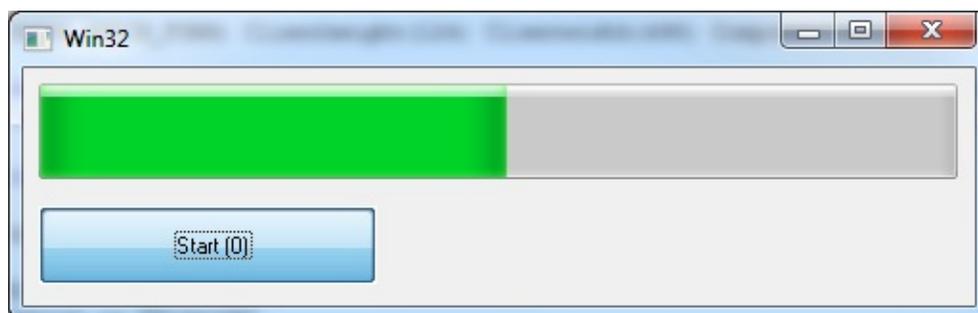
UpdateDisplay

Win32 では長い実行プロセスの間に画面を強制的にリフレッシュするよう制御する際に UpdateDisplay メソッドを呼び出すことができます。Win32 ランタイムは個別の制御を特定することができ、実際に UI の一部を更新することができました。

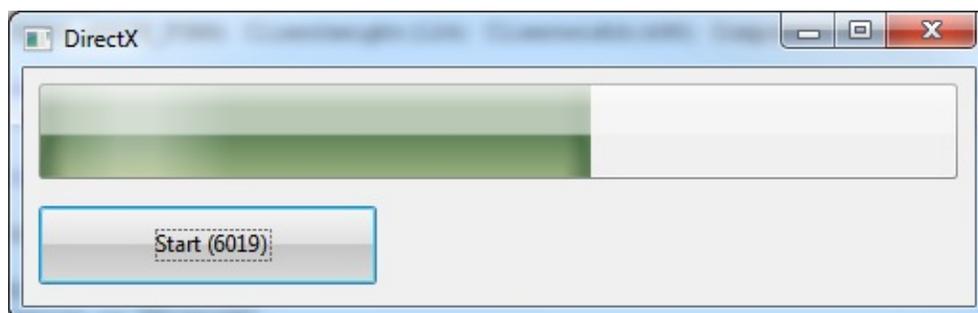
しかし、DirectX ランタイムは違う方式で動き、この方法は使用することができません。UpdateDisplay はフォーム全体の更新を引き起こします。

ほとんどの場合、これはさほど重要なことではありませんが、UpdateDisplay が繰り返し呼び出される場合には、明らかなパフォーマンスの低下を引き起こします。

この状況は、プログレス・バーを使用している際によく発生します。プログレス・バーは自動的に UpdateDisplay を使って最新の値を反映します。下の例では、簡単なループが実行され、プログレス・バーとスタートボタンのキャプションがその度に更新されます。



上の Win32 では、スタートボタンは更新されません。UpdateDisplay はプログレス・バー固有のものになります。しかし、DirectX では、フォーム全体が更新されます。



このフォームのコードは、このドキュメントの「サンプルソース」のセクションで確認することができます。

この状況に対応するために、プログレス・バーを更新したり、UpdateDisplay を繰り返しのたびに実行するのではなく、簡単な検査を追加することで、10回に1回だけ更新がかかるようにすることができます。

サンプルソース

[デフォルトの見た目](#)

[透明性と不透明性](#)

[マウスイベント](#)

[Win32 と DirectX \(ActiveX とグラフ\)](#)

[UpdateDisplay](#)

デフォルトの見た目

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(119) Clientwidth(336) Componentversion(1)
Height(157) Left(175) Top(215) Width(352)
Define_Com Class(#EMPNO.Visual) Name(#EMPNO) Componentversion(1) Displayposition(1) Left(8)
Marginleft(130) Parent(#COM_OWNER) Tabposition(1) Top(8)
Define_Com Class(#SURNAME.Visual) Name(#SURNAME) Componentversion(1) Displayposition(2)
Left(8) Marginleft(130) Parent(#COM_OWNER) Tabposition(2) Top(32)
Define_Com Class(#GIVENAME.Visual) Name(#GIVENAME) Componentversion(1) Displayposition(3)
Left(8) Marginleft(130) Parent(#COM_OWNER) Tabposition(3) Top(56)
Define_Com Class(#PRIM_PHBN) Name(#OK) Buttondefault(True) Caption('&OK') Displayposition(4)
Left(164) Parent(#COM_OWNER) Tabposition(4) Top(88)
Define_Com Class(#PRIM_PHBN) Name(#Cancel) Buttoncancel(True) Caption('&Cancel')
Displayposition(5) Left(252) Parent(#COM_OWNER) Tabposition(5) Top(88)
Evroutine Handling(#Com_owner.CreateInstance)
Case (#sys_appln.RenderStyle)
When (= DirectX)
#Com_owner.Caption := "DirectX"
When (= Win32)
#Com_owner.Caption := "Win32"
Endcase
Endroutine
End_Com
```

透明性と不透明性

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(114) Clientwidth(522) Height(152) Left(106)
Top(204) Width(538)
Define_Com Class(#PRIM_PANL) Name(#Details) Displayposition(1) Height(108) Left(15)
Parent(#COM_OWNER) Tabposition(2) Tabstop(False) Top(13) Width(338)
Define_Com Class(#PRIM_PANL) Name(#Address) Displayposition(2) Enabled(False) Height(108)
Left(15) Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(13) Width(338)
Define_Com Class(#PRIM_PHBN) Name(#MoveToFront) Caption('Show Address') Displayposition(3)
Left(360) Parent(#COM_OWNER) Tabposition(3) Top(8) Width(153)
Define_Com Class(#EMPNO.Visual) Name(#EMPNO) Componentversion(1) Displayposition(1)
Height(20) Left(8) Parent(#Details) Tabposition(1)
Define_Com Class(#SURNAME.Visual) Name(#SURNAME) Componentversion(1) Displayposition(2)
Height(20) Left(8) Parent(#Details) Tabposition(2) Top(24)
Define_Com Class(#GIVENAME.Visual) Name(#GIVENAME) Componentversion(1) Displayposition(3)
Height(20) Left(8) Parent(#Details) Tabposition(3) Top(48)
Define_Com Class(#ADDRESS1.Visual) Name(#ADDRESS1) Componentversion(1) Displayposition(1)
Height(20) Left(8) Parent(#Address) Tabposition(1) Width(300)
Define_Com Class(#ADDRESS2.Visual) Name(#ADDRESS2) Componentversion(1) Displayposition(2)
Height(20) Left(8) Parent(#Address) Tabposition(2) Top(24) Usepicklist(False) Width(300)
Define_Com Class(#ADDRESS3.Visual) Name(#ADDRESS3) Componentversion(1) Displayposition(3)
Height(20) Left(8) Parent(#Address) Tabposition(3) Top(48) Width(300)
Define_Com Class(#POSTCODE.Visual) Name(#POSTCODE) Componentversion(1) Displayposition(4)
Height(20) Left(8) Parent(#Address) Tabposition(4) Top(72) Usepicklist(False) Width(249)
Evroutine Handling(#MoveToFront.Click)
If (#Details.DisplayPosition <> 1)
#Details.DisplayPosition := 1
#Details.enabled := True
#Address.enabled := False
#MoveToFront.Caption := "Show Address"
Else
#Address.DisplayPosition := 1
#Details.enabled := False
#Address.enabled := True
#MoveToFront.Caption := "Show Details"
Endif
Endroutine
End_Com
```

マウスイベント

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(306) Clientwidth(462) Componentversion(1)
Height(344) Left(103) Top(200) Width(478)
Define_Com Class(#PRIM_Trvw) Name(#List) Columnbuttonheight(19) Componentversion(2)
Displayposition(1) Fullrowselect(True) Haslines(False) Height(261) Keyboardpositioning(SortColumn)
Left(8) Linesatroot(False) Parent(#COM_OWNER) Tabposition(1) Top(40) Viewstyle(UnLevelled)
Width(444)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Displayposition(1) Level(1) Parent(#List)
Source(#EMPNO) Width(27)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Displayposition(2) Level(2) Parent(#List)
Source(#SURNAME) Width(33)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_3) Displayposition(3) Level(3) Parent(#List)
Source(#GIVENAME) Width(40)
Define_Com Class(#PRIM_SPBN) Name(#Delete) Caption('Delete') Displayposition(2) Left(8)
Parent(#COM_OWNER) Tabposition(2) Top(8) Width(137)
Evroutine Handling(#Com_owner.CreateInstance)
Case (#sys_appln.RenderStyle)
When (= DirectX)
#Com_owner.Caption := "DirectX"
When (= Win32)
#Com_owner.Caption := "Win32"
Endcase
Select Fields(#List) From_File(pslmst)
Add_Entry To_List(#List)
Endselect
Endroutine
Evroutine Handling(#Delete.Click)
If (#List.CurrentItem *IsNot *null)
Dlt_Entry Number(#List.CurrentItem.Entry) From_List(#List)
Endif
Endroutine
End_Com
```

Win32 と DirectX (ActiveX とグラフ)

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(574) Clientwidth(965) Componentversion(1)
Height(612) Left(112) Top(184) Width(981)
Define_Com Class(#PRIM_PANL) Name(#Panel) Displayposition(1) Height(481) Horizontalscroll(True)
Left(80) Parent(#COM_OWNER) Style(#Style_1) Tabposition(1) Tabstop(False) Top(56)
Verticalscroll(True) Width(793)
Define_Com Class(#VA_WEBCTL.WebBrowser) Name(#Browser) Displayposition(1) Height(600)
Left(0) Parent(#Panel) Tabposition(1) Top(0) Width(775)
Define_Com Class(#PRIM_VS.Style) Name(#Style_1) Backgroundbrush(#RadialBrush_1)
Define_Com Class(#PRIM_VS.BrushColors) Name(#BrushColors_1)
Define_Com Class(#PRIM_VS.BrushColor) Name(#BrushColor_1) At(25) Color(255:255:255)
Parent(#BrushColors_1)
Define_Com Class(#PRIM_VS.BrushColor) Name(#BrushColor_2) At(100) Color(32:155:204)
Parent(#BrushColors_1)
Define_Com Class(#PRIM_VS.RadialBrush) Name(#RadialBrush_1) Colors(#BrushColors_1)
Originleft(100) Origintop(100) Radiusleft(125) Radiustop(125)
Evroutine Handling(#Com_owner.CreateInstance)
Case (#sys_appln.RenderStyle)
When (= DirectX)
#Com_owner.Caption := "DirectX"
When (= Win32)
#Com_owner.Caption := "Win32"
Endcase
Endroutine
Evroutine Handling(#Com_owner.initialize)
#Browser.Navigate( www.lansa.com )
Endroutine
End_Com
```

UpdateDisplay

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(124) Clientwidth(498) Componentversion(1)
Height(162) Left(261) Top(195) Width(514)
Define_Com Class(#PRIM_PGBR) Name(#ProgressBar) Displayposition(1) Left(8)
Maximumvalue(10000) Minimumvalue(0) Parent(#COM_OWNER) Tabposition(1) Top(8) Value(1)
Width(481)
Define_Com Class(#PRIM_PHBN) Name(#Start) Caption('Start (0)') Displayposition(2) Height(41)
Left(8) Parent(#COM_OWNER) Tabposition(2) Top(72) Width(177)
Evroutine Handling(#Com_owner.CreateInstance)
Case (#sys_apln.RenderStyle)
When (= DirectX)
#Com_owner.Caption := "DirectX"
When (= Win32)
#Com_owner.Caption := "Win32"
Endcase
Endroutine
Evroutine Handling(#Start.Click)
#ProgressBar.value := 0
Begin_Loop To(10000)
#ProgressBar.value += 1
#Start.Caption := ("Start (&1)").Substitute( #ProgressBar.Value.Asstring )
End_Loop
Endroutine
End_Com
```

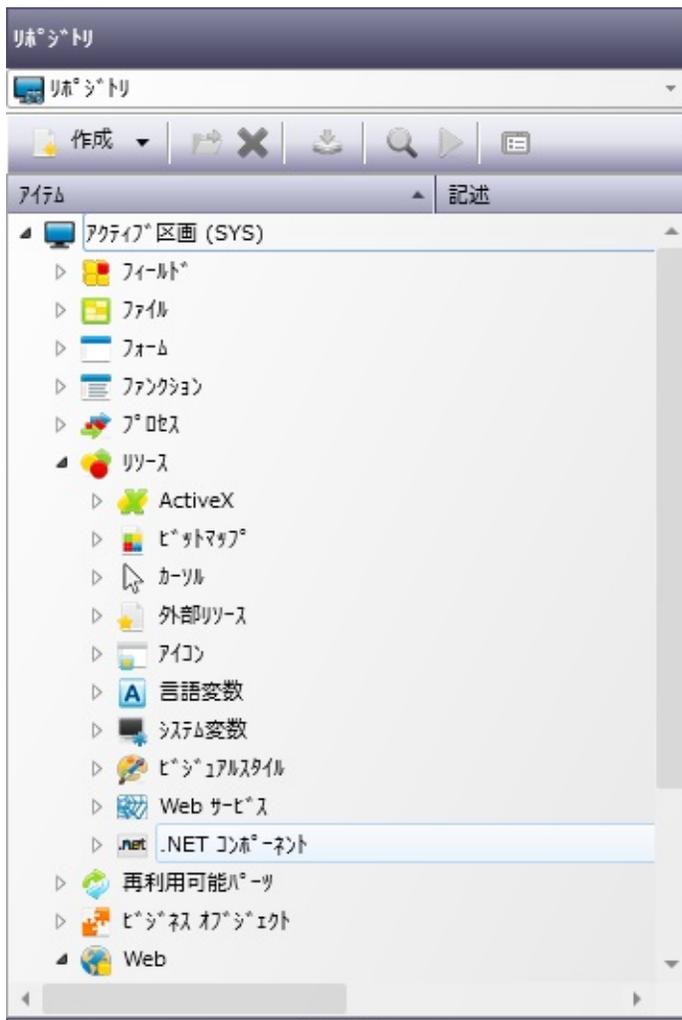
.NET コンポーネントを使用する機能

LANSA のプログラムで .NET のユーザーインターフェースや非ユーザーインターフェース・コンポーネントを使用することができます。

- グラフ
- ボタン
- バーコードスキャナ
- デバイス固有のインターフェース

リポジトリにサードパーティの .NET コントロールを登録し、RDMLX で用意されているイベント、メソッド、プロパティを使用することができます。

詳細については、『Visual LANSA 開発者ガイド』の「[.NET コンポーネント](#)」を参照してください。



モバイルのためのWAM

jQuery Mobile TSP は jQuery Mobile のタッチ対応 web フレームワークに基づいた、モバイルデバイス用の機能拡張された HTML 5 のソリューションを提供します。



jQuery Mobile は全ての一般的なモバイルデバイスのプラットフォーム向けの、jQuery 及び jQuery UI ファンデーション上に構築する、統合されたHTML5ベースのユーザーインターフェースシステムです。軽量化されたコードで、常に機能拡張がされており、フレキシブルで簡単にテーマに合わせた構成を行うことができるデザインを提供しています。

jQuery Mobile テクノロジーサービスは LANSIA IDE の中でプロパティを編集することで構成するドラッグ & ドロップ可能なウェブレットと共に稼働する JavaScript ライブラリを包んだものです。

サンプルのアプリケーションやモバイルアプリのデザインの骨組を作成するウィザードを使用して、モバイル WAM を作成することができます。



「LANSA Web モバイル アプリケーション ウィザード」も参照してください。

UNICODEを使用した国際化

LANSA は Unicode のフルサポートを提供します。

- データベース・テーブルはカラムにおいて UNICODE データタイプをサポートしていません。
- IDE は Unicode を使用して全ての複数言語テキストを表示するようになりました。ですので、実行したときと同様に全てのテキストが全ての言語で見ることができます。
- RDMLX コンポーネントやファンクションでの Unicode フルサポート。
- ユーザーインターフェース(フォーム、WAM とも) Unicode をサポートします。
- 使用言語に関わらず WAM で UTF-8 を出力します。
- IBM i 外部ファイルでの UTF-8 (CCSID 1208) サポート。
- Unicode の文字列組み込み関数を使用してデータの整合性を図ります。(AsNativeString および AsUnicodeString)
- Unicode フィールドタイプ Nchar 及び Nvarchar :

The screenshot shows a dialog box titled "新しいフィールド" (New Field). It contains the following fields and controls:

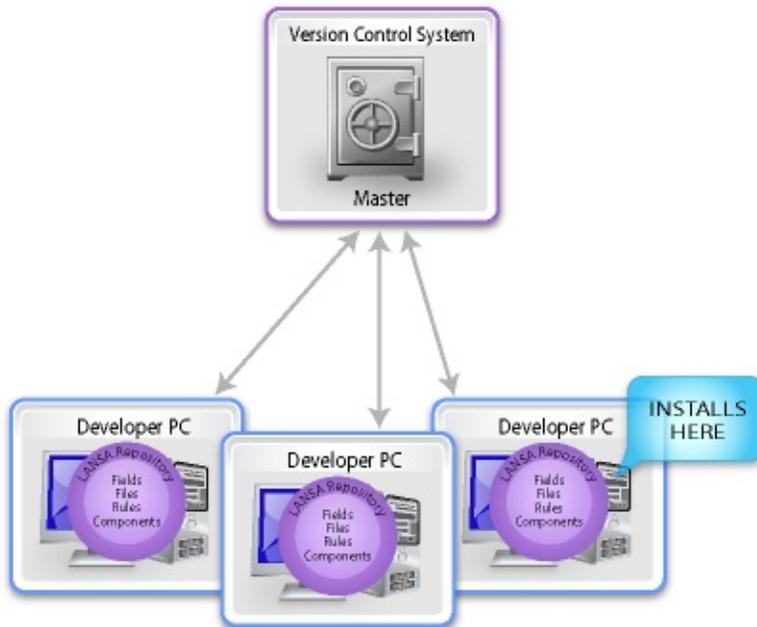
- 名前 (Name): Name
- 記述 (Description): Name
- フィールドタイプ* (Field Type*): NChar (highlighted with a red box)
- フィールド桁数 (Field Length): 10
- 小数点桁数 (Decimal Length): [empty]
- 参照フィールド* (Reference Field*): ...
- 識別子 (Identifier): NAME
- RDMLX使用可能 (RDMLX Useable):
- Buttons: 作成(C) (Create), キャンセル(N) (Cancel)
- Options: アイテムで開く (Open with item), 閉じる (Close)

Unicode の導入により、組み込み関数の規則にいくつかの変更が発生しました。変更については、技術解説書の「[組み込み関数の規則](#)」を参照してください。

バージョン管理システムサポート

Visual LANSA はIBM i をマスターとする代わりに、バージョン管理システムをマスターリポジトリとして使用することをサポートするようになりました。このシステム構成は、VCS マスターと呼ばれます。

SELECTED SCENARIO



各開発者はサンドボックス化され、VCS にチェックインされるまで他の開発者の仕事を妨げません。どんな VCS も使用することができますが、VCS に関する深い知識が必須となります。

VCS マスターは VCS を使用する予備知識を持つ Windows 開発者に使用されるよう設計されています。VCS を使用して、他の Windows 開発環境、例えば Visual Studio などと一緒に使用するのに似ています。

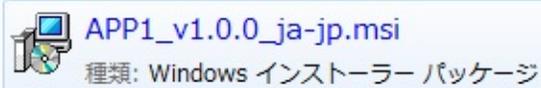
分岐、マージ、ソース比較、ラベル付与などの全てのバージョン管理機能は、ソースコードの管理をよりしやすくしますが、そのためには、入念な計画とルール付けが必要になります。

VCS はソースコードへのアクセスを管理するため、LANSA のタスク追跡や LANSA のセキュリティは無視され、ソースコードの編集スタンプは使用することができません。

詳細は、マニュアルを参照してください。

配布

LANSA アプリケーションは配布のために標準的な MSI パッケージとしてパッケージ化されるようになりました。



SCCM (インストール管理ソフトウェア) と統合した MSI は、複数言語のインストールに対してよりよいサポートを提供し、より目的にあうようにバージョン更新を可能にするよう構成可能なものとなります。

詳しくは、「[配布機能の新機能](#)」を参照してください。

ロングネーム

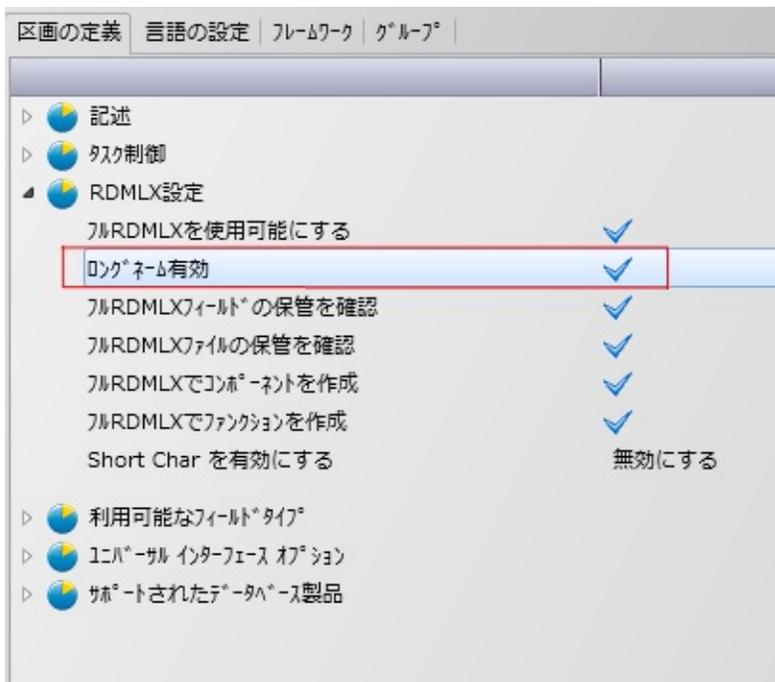
サードパーティとの統合や内容がわかるような名前をつけることができるように、ロングネームが導入されました。

いくつかの LANSA オブジェクトタイプが、RDMLX 区画で従来の10文字より長い名前で参照することができます。参照することができるオブジェクトタイプは以下のとおりです。

- フィールド
- ファイル
- 論理ビュー
- コンポーネント
- プロセス
- ファンクション
- WAM

ロングネームは Visual LANSA の中で、これらの LANSA オブジェクトにのみ受け渡されます。

ロングネームの使用は、RDMLX の区画レベルで設定します。



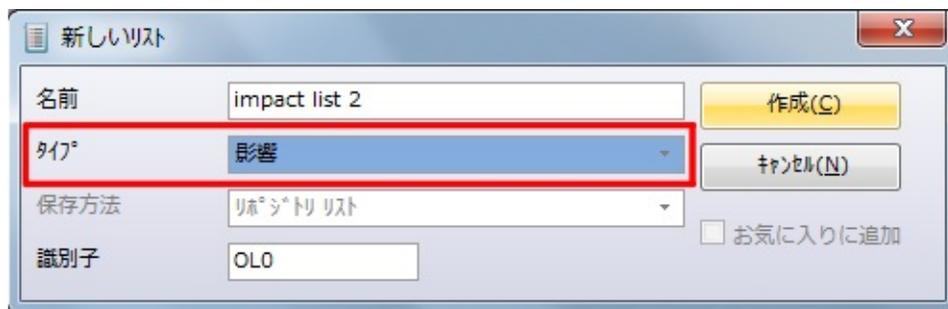
RDMLX 機能拡張

RDMLX では、以下のような機能が強化されています。

- 新しい組み込み関数と基本コンポーネント
- 新しい言語コマンド (GET/SET)
- 32000行のコード - コンパイラの制限
- 再帰スタックが 50 から 250 に増加
- インスタンスのコンストラクター
- インライン構築の *NEW 演算子
- 独自のメソッドへのカテゴリの割り当て

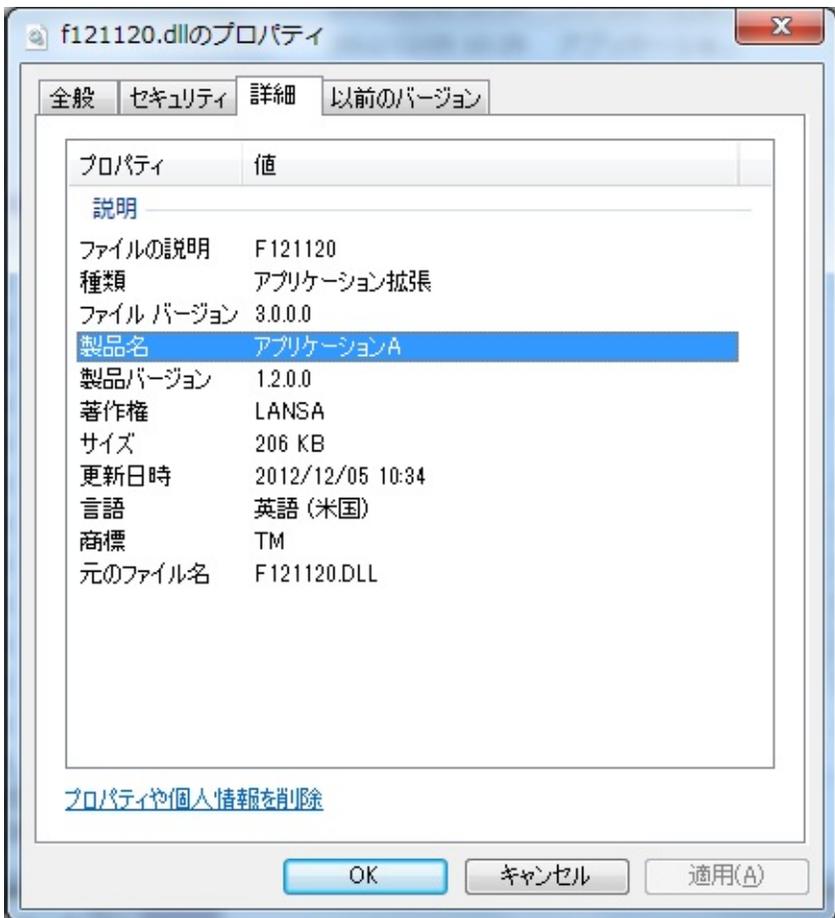
Windows を中心とした開発

- 影響分析などの保守・管理タスクが Visual LANSA で行えるようになりました。



詳しくは「[影響分析](#)」を確認してください。

- テーブルの ID 行の使用により、RPTH ファイルを使用する必要がなくなりました。
- 事後デバッグ機能が拡張されました。LANSA 実行ファイルはアクセスバイオレーションなど制御できない例外が発生した場合に自動的にダンプファイルを生成するようになりました。これによりそれらの稀な問題を解決するのにかかる時間を短縮し、問題を再現するのに難しいプログラムの状態をより簡単に記録することができます。
- 製品名、製品バージョン、ファイルバージョン、著作権などのバージョン情報をコンパイル時に LANSA オブジェクトに含めることができます。これらの情報は DLL のプロパティで目視することができます。



影響分析

影響リストを使用すると、特定のオブジェクトに関連するオブジェクトを見つけることができます。それにより、計画された変更の影響を見積もることができます。



影響というリストタイプを使用し、リストに含めたいオブジェクトを選択し、各オブジェクトタイプについて特定のもしくは共通のフィルターを指定して候補を絞ることができます。

例えば、共通のフィルターには名前全部、または一部を検索するように設定することが考えられます。特定のフィルターには、オブジェクトの各フィールドについて、アスタリスクのワイルドカードあり、またはなしで *Like*、*EQ* (等しい)、*GT* (より大きい)、*LT* (より少ない)、*LE* (以下) の演算子を含めることができます。



後で再度使用することができるように、リストは静的リストとして保存、または Excel ファイルとして保存することができます。

影響リストの実行

影響リストを実行するには、影響リストのツールバーで[実行]のボタン

をクリックしてください。

[実行]のボタンをクリックすると、同時に影響リストを保存します。

結果の確認

結果は影響分析の出力ビューに表示されます。



オブジェクト	記述	現在の処理	開始	終了
終了した	Impact List Search MyList	Search Ended - Found 6 Results	2012/12/04 18:42:34	2012/12/04 18:42:46
DXDOCS	Store documents command hand...			
FPDOC	Documents file			
FPNOTE	Notes file			
PSLEVENT	Personnel Event Log			
PSLIMG	Personnel Images			
PSLTIMES	Personnel time sheets			

エクスポート

出力ビューで、ジョブのヘッダーをクリックしてください。すると、影響分析によって生成されたオブジェクトのリストをエクスポートすることができます。

a. Excel にエクスポート

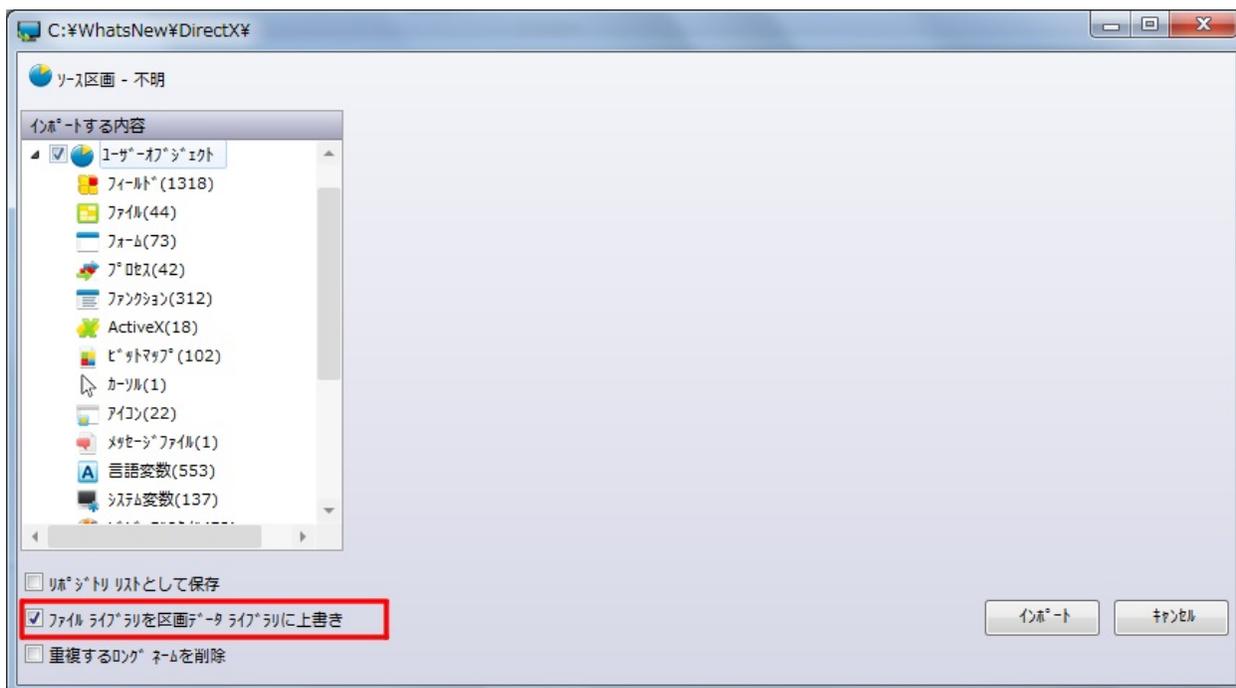
ジョブのヘッダーを選択したまま、[Excel にエクスポート]のアイコンをクリックしてください。LANSA オブジェクトのリストが CSV ファイルへエクスポートされます。

b. 静的リストにエクスポート

ジョブのヘッダーを選択したまま、[静的リストにエクスポート]のアイコンをクリックしてください。ダイアログが表示され、名前とリストのタイプを選択することができます。

インポート・配布時のコレクション/ライブラリの変更

ファイルライブラリやコレクションをインポートや配布時に変更してデフォルトの振る舞いを上書きすることができます。



デフォルトでは OAM は区画のデフォルトのファイルライブラリに従って名前を付けられ、区画の execute ディレクトリのサブディレクトリに作成されます。例えば、DEM 区画のデフォルトの区画ファイルライブラリが DC@DEMOLIB の場合、OAM は X_DEMDC@DEMOLIB\EXECUTE ディレクトリに作成されます。ファイルライブラリが区画のデフォルトライブラリと異なる場合、Visual LANSA IDE は OAM を区画の execute ディレクトリに置きます。

ファイルを配布する際のデフォルトの振る舞いは、対象区画のデフォルトのファイルライブラリを使用するよう、SQL テーブルと OAM を変更します。

ファイルの機能拡張

IBM i 外部ファイルは以下をサポートするようになりました。

- Binary および Varbinary フィールドタイプ
- UTF-8 (CCSID 1208)

Boolean はリポジトリのフィールドタイプとしてサポートされていません。

インストールと開発

新しいインストールと開発の機能が追加され、操作性と開発者の生産性を向上します。

サポート対象のプラットフォーム

LANSA のサポート Web サイトの [サポート対象のバージョン](#) のページ（英語）をご確認ください。

アップグレード・パス

全てのバージョン 12 のシステムは直接バージョン 13 にアップグレードすることができます。

バージョン 11(SP5) のシステムはまずバージョン 12 にアップグレードしなくてはなりません。

チェックイン・チェックアウトに関する注意

バージョン 13 にアップグレードすると、全てのオブジェクトのリポジトリの状態はチェックアウトされていませんに設定されます。これは、アップグレード前に全ての区画のオブジェクトへの全ての変更が IBM i にチェックインされなければならないということを意味しています。

アップグレード後、オブジェクトを PC にチェックアウトする際、ロックがかかり、他の PC はそのオブジェクトを修正することができません。オブジェクトをロックしたくない場合には、読み取り専用でチェックアウトしてください。Visual LANSA エディタにも IBM i にもロックを解除するオプションがあります。他の開発者がオブジェクトにアクセスできるようにするのに使用できます。



Visual LANSA フレームワーク

バージョン 13 の機能

このバージョンのフレームワークにより、DirectX のユーザーインターフェースを含めた LANSA バージョン 13 の新機能を使用することができます。

カスタマイズされたクイック検索

[クイック検索] ボックスは VLF ウィンドウの右上に表示されるダイアログです。

現在の振る舞いは、全てのビジネスオブジェクトのキャプションのリストを検索することです。これを上書きすることができるようになり、ユーザーは指定した値のリストを検索することができます。

ユーザーがひとつの値を選んだ場合に、何を行うか制御することができます。通常はビジネスオブジェクトへの切り替え、ビジネスオブジェクトのインスタンスリスト・エントリへの切り替え、ビジネスオブジェクトへのコマンドハンドラーへの切り替えを行います。

必要があれば、検索値のリストを再構築するように合図を送ることもできます。

モニター間の切り替えボタン

ユーザーが複数のモニター間を切り替えられるようにボタンが追加されました。ボタンはフレームワークウィンドウの左下にあります。

自動コマンドハンドラー浮動機能

ユーザーがフルサイズのインスタンスリストとフルサイズのコマンドハンドラーを同時に見る必要がある場合の新しい機能がフレームワークで使用可能になりました。

この機能では、インスタンスリストがクリックまたはダブルクリックされた際に、コマンドハンドラーを自動的に違うウィンドウとして浮動させます。

ユーザーが2つのモニターを使用している場合には、コマンドハンドラーは自動的にもうひとつのモニターへと浮動させることもできます。

これにより、元のウィンドウにフルサイズのインスタンスリストを残すことができ、ユーザーはコマンドハンドラーのウィンドウのサイズを変更することができます。

インスタンスリストのポップアップパネル・ヒント

フレームワークが Direct-X モードで実行されている場合、ユーザーがインスタンスリストのエントリー上にマウスを持っていった際にポップアップパネルを表示させることができます。このパネルを使って、その項目のコマンドハンドラーを開くことなく、ユーザーに項目の概要を簡単に伝えることができます。

エンドユーザーは、ポップアップが必要でない場合には、インスタンスリストを右クリックすることで、この機能を使用できなくすることができます。

VLF-WIN の小さな改善点

ユーザーがツリービューのインスタンスリストのクラスター項目をクリックした際、Visual ID1 と Visual ID2 が使用可能です。従来は、キーを識別する項目のみ使用可能でした。

日付のインスタンスリスト項目にブランクの値が追加された場合、これまでのインスタンスリストのエントリーの値ではなく、ブランクが表示されます。

ユーザーが複数のビジネスオブジェクトに適用されるコマンドを選択した際のビジネスオブジェクトの並び替え機能が改善されました。

LANSA Integrator

バージョン 13 での LANSAs Integrator の機能拡張には以下が含まれます。

- LANSAs フィールドのロングネームのサポート
- BLOB/CLOB サポート
- Unicode フィールドのサポート
- JSON ファイルの読み書きを可能にする JSONBindFileService
- SMTP 及び POP3 サービスへの暗黙的・明示的な SSL/TLS 接続のサポート

LANSA バージョン 13の新機能

ライセンス	新しいバージョン 13 のライセンスが、IBM i / Windows共に必要となります。
Microsoft DirectX ユーザーインターフェース	Microsoft DirectX ユーザーインターフェースが、Visual LANSАの高いエンドユーザー・エクスペリエンスとデータのよりリッチなビジュアルイゼーションを提供します。
.NET コンポーネントを使用する機能	LANSAのプログラムで .NET のユーザーインターフェースや非ユーザーインターフェース・コンポーネントを使用することができます。
モバイルのためのWAM	モバイルアプリケーションのためのWebアプリケーション・モジュール (WAM)を簡単に作成することができます。
UNICODEを使用した国際化	LANSA はフル Unicode サポートを提供します。
バージョン管理システムサポート	サードパーティのバージョン管理ツールへのLANSAのバージョン管理のインターフェースの第一段階が使用可能です。
配布	配布のために標準的なMSIパッケージとしてLANSAアプリケーションをパッケージ化することができます。
ロングネーム	ロングネームを使用することで、サードパーティとの統合や内容がわかるような名前をつけることができます。
RDMLX 機能拡張	新しい組み込み関数や基本コンポー

Windows を中心とした開発

ネット、言語のGET/SETコマンドなどが使用可能になりました。

様々な機能拡張がLANSAになされ、それにより、更にWindows プラットフォーム上での開発が中心となります。

ファイルの機能拡張

IBM i 外部ファイルはUnicode、binary 及び varbinary フィールドをサポートするようになりました。

インポート・配布時のコレクション/ライブラリの変更

ファイルライブラリやコレクションはインポートや配布の際に上書きできるようになりました。

インストールと開発

新しいインストールと開発の機能が追加され、操作性と開発者の生産性を向上します。

Visual LANSA フレームワーク

新しいバージョンのフレームワークでは、DirectXのユーザーインターフェースを使用できるようになりました。

LANSA Integrator

最新のLANSA Integrator がバージョン 13 と共に出荷されます。

エディション日付：2012年10月30日

© 2012 LANSA