



Migration from uniPaaS V1.x to Magic xpa 3.x

Self-Paced Tutorial

Book ID: UTMU19MA3

Edition: 1.00, March 2015

Course ID: UCMG19MA3

Magic University Official Courseware

The information in this manual/document is subject to change without prior notice and does not represent a commitment on the part of Magic Software Enterprises Ltd.

Magic Software Enterprises Ltd. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose.

The software described in this document is furnished under a license agreement. The software may be used or copied only in accordance with the terms and conditions of the license agreement. It is against the law to copy the software on any medium except as specifically allowed in the license agreement.

No part of this manual and/or databases may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or information recording and retrieval systems, for any purpose other than the purchaser's personal use, without the prior express written permission of Magic Software Enterprises Ltd.

All references made to third-party trademarks are for informational purposes only regarding compatibility with the products of Magic Software Enterprises Ltd.

Unless otherwise noted, all names of companies, products, street addresses, and persons contained herein are part of a completely fictitious scenario or scenarios and are designed solely to document the use of Magic xpa.

Magic® is a registered trademark of Magic Software Enterprises Ltd.

Btrieve® and Pervasive.SQL® are registered trademarks of Pervasive Software, Inc.

IBM®, Topview™, System i5/iSeries™, System i™, IBM i™, pSeries®, xSeries®, RISC System/6000®, DB2®, and WebSphere® are trademarks or registered trademarks of IBM Corporation.

Microsoft®, FrontPage®, Windows™, WindowsNT™, ActiveX™, and Windows Mobile are trademarks or registered trademarks of Microsoft Corporation.

Oracle® and OC4J® are registered trademarks of the Oracle Corporation and/or its affiliates.

Linux® is a registered trademark of Linus Torvalds.

UNIX® is a registered trademark of UNIX System Laboratories.

GLOBEtrotter® and FLEXIm® are registered trademarks of Macrovision Corporation.

Solaris™ and Sun ONE™ are trademarks of Sun Microsystems, Inc.

HP-UX® is a registered trademark of the Hewlett-Packard Company.

Red Hat® is a registered trademark of Red Hat, Inc.

WebLogic® is a registered trademark of BEA Systems.

Interstage® is a registered trademark of the Fujitsu Software Corporation.

JBoss™ is a trademark of JBoss Inc.

Systinet™ is a trademark of Systinet Corporation.

Android is a trademark of Google Inc.

BlackBerry® is a registered trademark of Research in Motion Limited.

iPod, iPad, iPhone, iTunes, and Mac are registered trademarks of Apple Inc.

Portions Copyright © 2002 James W. Newkirk, Michael C. Two, Alexei A. Vorontsov or Copyright © 2000-2002 Philip A. Craig
Clip art images copyright by Presentation Task Force®, a registered trademark of New Vision Technologies Inc.

This product uses the FreedImage open source image library. See <http://freeimage.sourceforge.net> for details.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>).

This product includes software developed by Computing Services at Carnegie Mellon University (<http://www.cmu.edu/computing/>). Copyright © 1989, 1991, 1992, 2001 Carnegie Mellon University. All rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

This product includes software that is Copyright © 1998, 1999, 2000 of the Thai Open Source Software Center Ltd. and Clark Cooper.

This product includes software that is Copyright © 2001-2002 of Networks Associates Technology, Inc All rights reserved.

This product includes software that is Copyright © 2001-2002 of Cambridge Broadband Ltd. All rights reserved.

This product includes software that is Copyright © 1999-2001 of The OpenLDAP Foundation, Redwood City, California, USA. All Rights Reserved.

All other product names are trademarks or registered trademarks of their respective holders.

Migration from uniPaaS V1.x to Magic xpa 3.x

March 2015

Copyright © 2015 by Magic Software Enterprises Ltd. All rights reserved.

Contents

Introduction	1
Seminar Prerequisites	1
About the Seminar	2
Magic Software University	2
How to Use this Guide	3
Exercises.....	3
Additional Materials.....	4
Major New Capabilities	5
Runtime Engine Using WinForms Controls.....	5
.NET Code and Controls	6
Form Designer Based on Visual Studio	6
Mobile Applications.....	6
Offline Applications	6
In Memory Data Grid Database	6
In Memory Data Grid Middleware.....	7
Migrating Your Project.....	9
Fonts	10
Colors	11
Migrating a uniPaaS V1.x Project.....	11
Converting an Application with Components	13
Importing an Application or Part of One.....	13
Suggested Methodology	14
Exercise.....	14
Summary	14
Magic xpa MDI	15
MDI and SDI	16
Multiple Document Interface (MDI)	16
Single Document Interface (SDI).....	16
MDI.....	17
Main Program Form	17
Application Properties	18

Application Icon	18
Application Caption.....	18
Modified Application Properties	19
Environment Properties	20
Status Bar	20
MDI Color.....	21
MDI Client Area	22
Exercise	22
Summary	22
UI Changes	23
Form Designer	24
Expression Editor	25
Form Properties.....	25
Startup Position	25
Form State Identifier	26
Control Properties	26
Top / Left	26
Height / Width	27
Style and Border Style.....	27
Horizontal Alignment	28
Z-Order Changes	28
Text Controls	28
RTF Controls.....	29
Edit Controls.....	29
Radio Button Controls.....	29
Static Controls	30
Table Controls	30
Image Controls	30
Push Button Controls.....	30
Choice Controls.....	31
Group Controls.....	31
Tab Controls	31

Tree Controls.....	31
Subform Controls.....	31
Unsupported Controls.....	32
ActiveX and OLE Controls.....	33
Summary	34
GUI Frames	35
Why Use Frames?.....	36
Using Frames	36
Automatic Refresh.....	38
Frame Name.....	39
Dynamic Display	39
Retain Focus.....	40
Splitter Conversion.....	40
Splitter Placement	42
Exercises.....	43
Summary	43
End User Functionality	45
Adding End User Functionality.....	46
Locate Row	46
Ranges	47
Sorting	48
Print Data.....	48
Unsupported Features.....	49
Exercise	50
Summary	50
Additional Changes	51
Task Behavior Changes.....	52
Task Properties	52
Task Modes	52
Control Verification.....	52
Changed Functions.....	52
Window Pulldown Menu	53
Environment Settings	53

Window Focusing.....	54
Magic xpa Runtime.....	54
Start Application.....	54
Internal Events	55
Encrypted SQLite Database Support.....	55
Summary	56
Solutions	57
Lesson 4 – GUI Frames.....	59
Travel Destinations.....	59
List of Deals	61
Customer Orders	62
Lesson 5 – End User Functionality	63
Changing the Print Data	63
Adding Print Functionality.....	63
Filtering Field Names	64

Introduction

Welcome to Magic Software University's **Migration from uniPaaS V1.x to Magic xpa 3.x** self-paced tutorial. We, at Magic Software University, hope that you will find this tutorial informative and that it will assist you in enhancing your knowledge of uniPaaS.

Seminar Prerequisites

Before you start with the seminar there are a number of prerequisites:

Development knowledge	Working knowledge of uniPaaS V1.x, preferably V1.9x.
-----------------------	--

The Studio and Runtime modules of Magic xpa are based on the .NET framework.

The following .NET framework requirements apply:

- To develop an application using the Magic xpa Studio, you must have .NET framework V4.0 (or above) installed on your machine.
- If you plan to run GigaSpaces on your server, you must have .NET framework V4 (or above) installed on the server.
- To run Online, Rich Client and Browser Client programs, you must have .NET framework V2.0 SP2 (or above) installed on your client machine.

When using .NET framework V2.0 SP2, you also need to install the Microsoft Visual C++ 2008 Redistributable. You can install them by running the **NetFx20SP2_x86.exe** and **VCRedist_x86.exe** files from the **Scripts\RIA** folder.

About the Seminar

The Magic Software University's **Migration from uniPaaS V1.x to Magic xpa 3.x** seminar is based on Magic xpa 3.0.

This seminar assumes that you have a working knowledge of uniPaaS programming. It is designed to help you improve your uniPaaS programming skills.

In this seminar you will learn about:

1. Major new capabilities in Magic xpa 3.x.
2. Migrating an application developed in uniPaaS V1.x.
3. The changes made to the Magic xpa MDI, which provide you with more control.
4. The changes in the user interface incurred as a result of the upgrade to .NET.
5. The new GUI Frames interface type.
6. The new and improved Form Designer and Expression Editor.
7. Using the new End User Functionality component.

By the end of the course you will:

- Acquire the skills and knowledge required to migrate a uniPaaS V1.x application to Magic xpa 3.x.
- Become familiar with the new functionality of Magic xpa.
- Comprehend the behavior changes between the versions and practice how to deal with them as a result of the migration process.
- Know what is no longer supported in Magic xpa.

Magic Software University

MSU offers various courses that may be of interest to you. To see the current offering, open the following link in your browser:

<http://www.magicsoftware.com/en/services/?catID=59&pageID=45&subPageID=756>

How to Use this Guide

In this self-paced tutorial, in each lesson you are first introduced to the subject matter and then there is an example implementing the theory you have just learned. The tutorial provides step-by-step solutions for the examples.

This guide also contains boxes with more information and tips. Here is what you may find:



This is a hint.



This is an important note.



This is a question for you.
Something to think about.

Exercises

At the end of most lessons there is an exercise. The solutions for the exercises are provided at the end of this book. Try to do each exercise on your own before looking at the solution.

Note that your solution may differ from the solution offered. This does not mean your solution is incorrect, as there are many ways to solve a problem.

A sample project is provided with this seminar. This is installed together with the seminar installation and can be found in the **Projects** folder. The name of the project is **Travel Agency**. Open this project when you start the seminar.

Programs are provided as a basis for some lessons. Each set of programs is located in its own folder in the Program repository.

Additional Materials

The seminar comes with the following materials:

- Seminar data – This is data, including projects, that you will need during the seminar's exercises. The seminar uses the SQLite database, which is a part of the Magic xpa installation. The seminar's SQLite database is provided with the installation. Templates, text files, XML files and images that are used during the seminar are also provided.

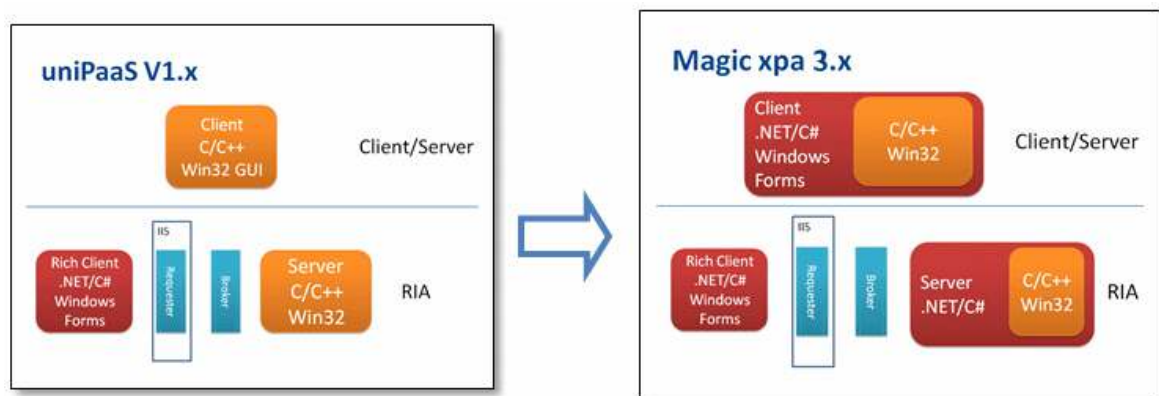
Lesson 1

Major New Capabilities

Runtime Engine Using WinForms Controls

Magic xpa has a new runtime engine for client/server and server-side applications that is integrated with the Microsoft .NET framework. This new runtime engine uses controls from the "Windows Forms" .NET library as the GUI front-end, and allows integration with any .NET class or third-party library quickly and easily. The .NET-based client was initially introduced in version 1.x with the Rich Client Runtime engine.

The Magic xpa Runtime engine is an extension of the uniPaaS V1.x Runtime engine. A new .NET front-end wraps the existing C++ engine, thus enabling both .NET and unmanaged functionality in a single runtime engine.



The .NET framework provides lots of functionalities, however it is not fully backward compatible. Most of the incompatibilities in the version were caused by this change.

.NET Code and Controls

With the full integration of .NET, it is now possible to use .NET functionality on Client/Server applications and also on the server side for Rich Client applications. This enables you to invoke .NET in a batch process on the server.

To benefit fully from these new functionalities, see the **Dot Net 3rd Party Samples** project included with Magic xpa 3.x.

Form Designer Based on Visual Studio

The new Visual Studio-based Form Designer offers an intuitive and user-friendly experience that makes designing forms even easier. By integrating with Visual Studio, the new Form Designer now offers an even more enhanced WYSIWYG experience.

Mobile Applications

You can develop mobile applications running on Android and iOS devices with the same development paradigm as you developed your own application. To benefit fully from these new functionalities, see the **Developing Mobile Applications** concept paper.

Offline Applications

You can develop Offline applications, which allow users to continue and be productive in areas with intermittent, limited or unavailable internet connectivity. While working offline, data is stored locally on a local database, and periodically, when internet connectivity resumes, you can synchronize it back to the server.

To benefit fully from these new functionalities, see the **Developing Offline Rich Client Applications** concept paper.

In Memory Data Grid Database

Magic xpa 3.x provides a new database gateway, the IMDG gateway, for storing and accessing data in the space. This gateway is similar to the other Magic xpa database gateways and enables complete access to information stored in the space.

GigaSpaces provides the opportunity to work with asynchronous persistency in what's referred to as Write Behind mode, which provides the ability to handle extremely large transaction workloads. The data is uploaded from a backend database and is updated to the space. GigaSpaces then does the work of updating the database server. The gateway allows access to the data that is connected to the application, while significantly improving performance and ensuring that data is not lost.

To benefit fully from these new functionalities, see the **Deploying Applications on GigaSpaces** concept paper in the *Magic xpa Help* or the **support** folder for additional information.

In Memory Data Grid Middleware

Magic xpa 3.0 provides an additional enhanced and robust architecture. The award winning XAP middleware from GigaSpaces is now supported as a messaging layer, which uses a distributed, self-healing and scalable architecture. These benefits directly enhance Magic xpa 3.0 projects. Existing projects upgraded to this version can be easily scaled to multiple processes and even to multiple machines (scale out) just by changing external configurations.

Note that the Magic Request Broker middleware is still supported as in the previous versions.

To benefit fully from these new functionalities, see the **Deploying Applications on GigaSpaces** concept paper.

Lesson 2

Migrating Your Project

When you receive a new major release of Magic xpa one of your first tasks is to migrate your current projects. Each new version adds exciting new features that you want to implement in your current projects to improve your product. However, some features that you currently use may no longer be available since the infrastructure was changed.

The migration of a project is a one-way process. You will learn about migrating your project in this lesson.

This lesson covers several topics including:

- Changes to fonts
- Changes to colors
- Converting an application
- Importing an application
- Converting an application with components
- Suggested methodology

Fonts

With the upgrade of the Magic xpa engine to .NET, some fonts and font properties are no longer supported because they are not supported by .NET.

Magic xpa includes the following changes:

- Font orientation for GUI forms is only supported for Label controls and column headers. You can still use orientation in your reports (forms with a class greater than zero) for use in PDFs and other forms of output.
- Only TrueTypes and OpenTypes are supported. These are generally files with a .ttf or .otf extension on the Windows operating system. As an example, MS Sans Serif is a regular font with an extension of .fon. It is not supported in a GUI form. Microsoft Sans Serif has a .ttf extension and is supported. The reason is that the runtime client is a .NET client, which does not support the out-dated fonts. The Checker will continue to issue errors for non-TrueType fonts until you replace them. For Output forms, you can use any font that your printer can display.

Your font file from the previous version may contain fonts that are no longer supported. It is recommended that you make the changes to the font file before you upgrade your project to the new version.

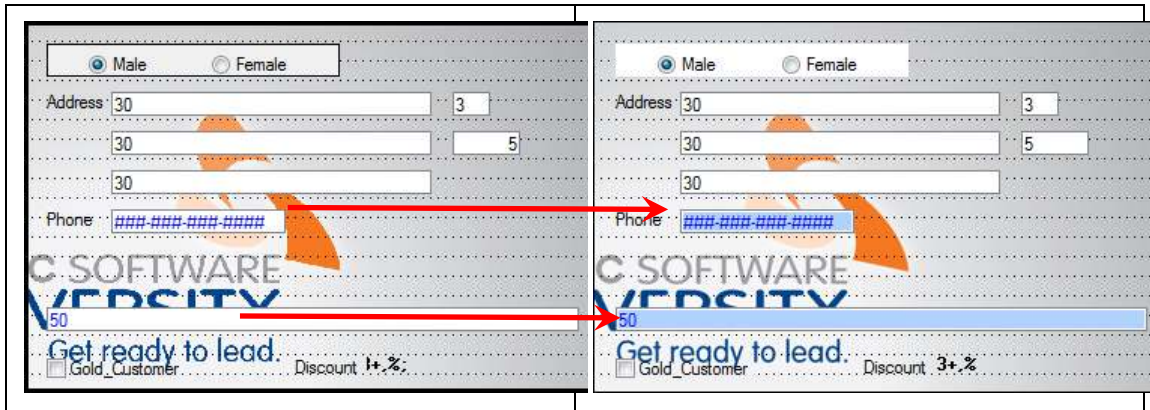
The course installation process sets the **Runtime Application Font Definition File** environment property to point to the font file created under the **Projects** directory for the course project.

1. Open the **Environment** dialog box from the **Options > Settings** menu.
2. Click the **External** tab.
3. Zoom into the Runtime Application Font Definition File property.
4. Park on the **Text Caption** entry and zoom into the **Font** dialog box. Select **Microsoft Sans Serif**.
5. Park on the **Text** entry and zoom into the **Font** dialog box. Select **Microsoft Sans Serif**.

When you make these changes to your own project it is good practice to search for where you used those fonts and check the effect it has on the look and feel of your program.

Colors

In V1, an Edit control with a 3D style displayed either a white or a grey background. The background color you selected was ignored. With the upgrade of the controls to .NET, the background color will now be visible. The image below shows an example of the change:



The left image shows Edit controls from a previous version and the image on the right shows the same program in Magic xpa 3.x. As you can see, the background color of two of the Edit controls is displayed.

This result may be what you were expecting. If this is not the desired result, you need to make the change to the color file in the previous version before you convert your application.

Migrating a uniPaaS V1.x Project

When you open a project that was developed in an earlier version, Magic xpa can open that version and make changes, upgrading the version to the current version on the fly.



This is normally a one-way process, so make a backup before opening your project in a newer version!

1. The course package includes a uniPaaS V1.9 project named **Travel Agency**. Copy it into the **Projects** folder of Magic xpa. You will be working with this project.
2. From Magic xpa 3.x, go to the **File** menu and select **Open Project**. You can also use the **Ctrl+O** shortcut.

3. Select the **Travel Agency** project.

When you open a project created in V1 .x, Magic xpa automatically opens the conversion utility.



The conversion utility upgrades your project to the new version. You will no longer be able to open your project in the previous version. The utility creates a backup if you require it. You have the choice whether or not to create one.

4. Click **Next** to continue.

Some of the Client/Server end user functionality, such as runtime range or locate is now performed using Magic programs, which are available in the End User functionality component. (This is the same component that was provided in previous versions for Rich Client only). You will learn more about this component in a later lesson. The conversion dialog box enables you to add the component to your project.

Once the conversion utility has upgraded your project, a log is created. This log provides you with information as to what was removed from your project. You will learn more about these changes during this course. The log is named **IMP.LOG** and you will find it in your project folder.

Here is an example of a log:

```
Converting project (14/02/13 14:54:25)
-----
Model 19 - Info - Tab control cannot be attached to a table control.
Model 52 - Info - Tab control cannot be attached to a table control.

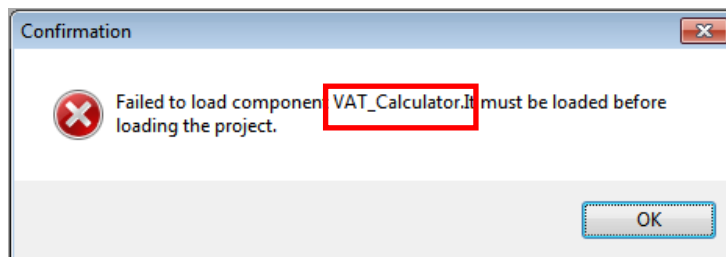
Program/Task 1 - Info - MDI form was created
Program/Task 4 - Warning - Splitter form changed to frameset. Manual logic change is required
Program/Task 7 - Warning - Splitter child form was changed. Manual logic change is required
Program/Task 9 - Info - Modifiable property for Image control is not supported and was removed.
Program/Task 9 - Info - Allow Parking property for Image control is not supported and was removed.
Program/Task 9 - Info - Tab Into property for Image control is not supported and was removed.
Program/Task 14 - Warning - ActiveX control is not supported. Manual logic change is required
Program/Task 16 - Warning - Splitter child form was changed. Manual logic change is required

General - Info - User State Persistency changed from Name to Yes/No. (14)
General - Info - Style and Border of the controls were changed. (47)
General - Info - The end-user functionality component was added to the project.
```

Converting an Application with Components

The conversion utility does not convert components. If your project has components, you need to upgrade the components first and then create a new ECF of the migrated component. This is important because the component may contain models that are used in the current project.

When you upgrade an application to Magic xpa, the conversion process checks the version of the components. If it finds that a component has not been converted, you will receive an error message:



Importing an Application or Part of One

Magic xpa enables you to import an application from a previous version or even a single program. When you import an application, Magic xpa automatically upgrades the application to the new version. The conversion dialog box will not appear. A conversion log is still created in the project folder.

If your previous version's application has components and you have not converted them, you will receive an error message similar to the one above.

Suggested Methodology

You already know how to upgrade an application. To successfully upgrade your application, here are some steps to consider.

In the previous version:

- Check the fonts used in V1.x and if necessary define a different font family. Remove orientation for fonts on GUI forms.
- Check the background colors of 3D controls. For the purpose of consistency you may decide to set the background color to white.
- Run the syntax checker on your V1.x application.
- Back up your V1.x application.

In Magic xpa 3.x:

- Convert components.
- Convert your project.
- View the conversion log.
- Execute the syntax checker. You need to execute this on models, menus and programs.
- You then need to make the manual changes.

Exercise

In this lesson, you imported an application that you will be using during this seminar.

As an exercise, a second application has been provided, **Chocolate**. This has a component named **Calculate**. Migrate the **Chocolate** project to Magic xpa 3.x.

No solution is provided for this exercise.

Summary

In this lesson, you learned about the migration process. Migrating a project to a newer version is an integral part of the life cycle of an application.

You learned how to migrate an application and what steps to take in the previous version to make the upgrade smoother. You learned about migrating an application that has components. You were provided with a suggested methodology or check list that you can implement when you start your own migration project.

In later lessons you will learn about the manual modifications you will need to perform in Magic xpa.

Lesson 3

Magic xpa MDI

In V1, in deployment, you displayed the entire frame containing the system bar, the status bar as well as the menus. If you did not want to display the menu, then you reset the pulldown menus. You had little control over the MDI. If you did not want to display the MDI, you defined an SDI application. You also needed to learn how to define an SDI application.

The current version gives you full control of what to display in the MDI: the size of the frame, the position, the status bar messages and other properties, all encapsulated in a single place – the Main Program.

This lesson covers several topics including:

- MDI frame
- Application properties
- Environment properties
- MDI color
- Status bar

MDI and SDI

Within the Windows operating system, a window that contains a title bar is sometimes referred to as a document. The Multiple Document Interface (MDI) and Single Document Interface (SDI) are methods for handling windows.

The MDI and SDI functionality enables an application to open and concurrently execute several tasks.

Multiple Document Interface (MDI)

This method uses a single primary window, which is the parent window, to visually contain a set of related child windows. Each of the child windows is a primary window, but it is constrained to appear only within the parent window. Child windows share the menu bar, status bar, toolbar and other parts of the parent window interface.

The MDI enables you to open more than one window simultaneously. The child windows automatically close when their parent window is closed. It is a user interface characterized by an open space that can hold one or more open windows, all enclosed in a single frame with one overhead menu. MDI is useful because one overhead menu can be used for any of the child windows, reducing clutter.

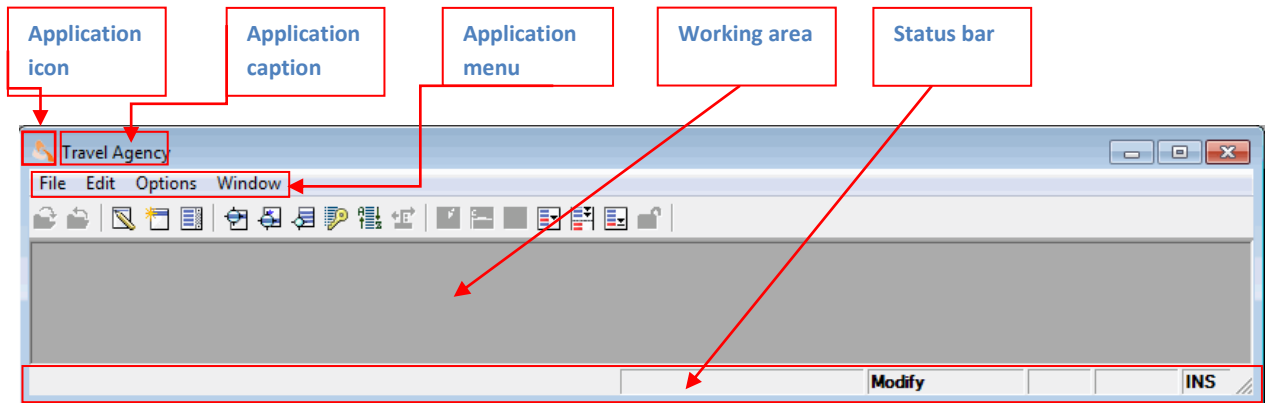
Single Document Interface (SDI)

The SDI window opens each document in its own primary window. When a new window is opened, the operating system places an entry on the taskbar, making it easy to access the content of this window.

An SDI form has its own menu, toolbar, and entry in the status bar. The desktop and taskbar provide the user with ways to manage the primary windows. Secondary modal windows are used to view information about objects in the primary window. The SDI is not constrained to a parent window. It is a "standalone" window, and is not part of the MDI. In many ways, it acts as if it were its own application, even though several SDI frames might be running under the same application.

MDI

The MDI comprises various components as you can see from the screen capture below. The screen capture was taken from an earlier version.



The application icon is in the top left section of the MDI and appears in the top left of each program's frame. The application caption is what appears both at the top of the frame as well as on the Windows task bar. The pulldown menu is visible together with the toolbar. Then you have the working area where all the programs are displayed, unless they are floating or SDI. At the bottom of the form, you have the status bar where the user receives error messages and can see the mode of the current program and other options.

In V1, you could manipulate some of these issues but the properties were spread around. In Magic xpa, the client/server runtime MDI window is no longer hard-coded into the runtime engine. Each application can define its own MDI window properties in the Main Program.

Main Program Form

The Main Program has a form. In V1, you used this form for global displays such as wallpaper, a logo, the date and others.

The Main Program's form works just as the other program forms do, except that they will show as long as the project is running, behind other task forms. You cannot have any data entry on the Main Program form, but you can have action controls, such as push buttons.

In V1, the Main Program's form was not displayed unless you implicitly set it to "Yes" by setting the **Task Properties->Interface-> Form->Open task window** property to **Yes**.

This behavior has changed in Magic xpa and by default the **Open task window** property is set to **Yes**. The form **Window Type** is automatically set to **MDI Frame**.

In this lesson, you will learn more about how this enhances your application.

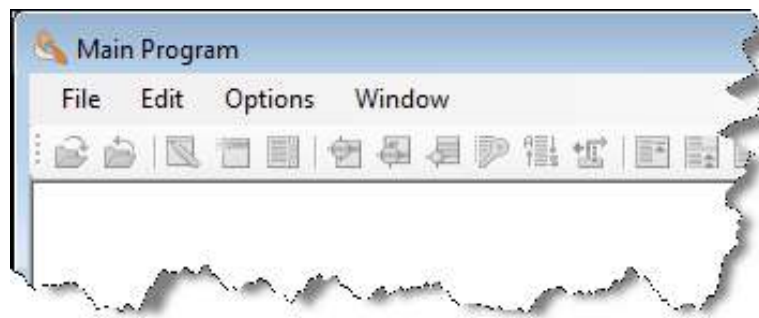
Application Properties

Application Icon

You can provide your own icon for your application. In V1, this option was available in the **Application Properties** dialog box. In Magic xpa, the **Icon File Name** property is now in the Main Program. As with any program, you set the program's icon in the **Task Properties** dialog box in the **Interface** tab. If you do not set an icon name, you will receive the default Magic xpa runtime icon.

Application Caption

The application caption is what appears at runtime as seen in the image on the previous page. In previous versions, the name that you provided when creating the project is the name that would appear during runtime. In Magic xpa, this is no longer the case. When you run the project, you will receive **Main Program** as the caption, as seen in the image below:



In V1, you could provide your own name for the application by setting a value in the **Caption** property in the **Application Properties** dialog box; thereby providing a more meaningful name to your application.

The **Caption** property is no longer a part of the **Application Properties** dialog box. It is now the **Form Name** of the Main Program. If you set the **Caption** property in V1, the conversion utility sets the **Form Name** to what was defined in the **Caption** property. You can keep the previous behavior by using the **AppName()** function. The **AppName()** function fetches the name you provided when you created the project. You implement this by setting the **Form Name** property to **AppName()**.

If you used a logical name in the **Caption** property you will need to define an expression using the **Translate ()** function.



It is good practice to provide a caption for your application.

Sometimes the project name is needed when you need to invoke the application externally, such as from the command line. As before, you can continue this behavior by setting the **Form Name** property to **AppName()**.

Modified Application Properties

As the Main Program form is the same as other forms, a number of properties from the **Application Properties** dialog box were moved to the form:

Application Property	Form Category	Form Property
Caption	Details	Form Name
System Pulldown Menu	SDI	Pulldown Menu

The **Main Display** task property in the Main Program enables you to define several MDI frame forms and dynamically display the required form at runtime.

The Main Program's **Task Properties** dialog box contains a new property called **Run project as**, which determines whether the Online or Rich Client MDI frame will be shown when running the project from the Studio (Ctrl+F7).

In V1, it was possible to show a Main Program form over the MDI and thus show static controls. This is currently not supported.

Environment Properties

Some MDI environment properties were removed from the INI and moved to the Main Program form or removed altogether. You need to update the values manually in the converted project:

INI Property	Form Category	Form Property	Comments
RfToolBarGUI	SDI	Display Toolbar	Part of the MDI.
MDIClientEdge			Not supported.
MDIClientImageFile	Appearance	Wallpaper	Not supported for Studio. This setting was moved into the Main Program's form as the Wallpaper property.
MDIClientImageStyle	Appearance	Wallpaper style	Partially supported. This setting was moved into the Main Program's form as the Wallpaper Style property.
DeploymentMode=SDI			Removed. This is part of the SDI application mechanism. To implement this functionality in Magic xpa, you need to simply hide the MDI window. To do this you simply set the Task Properties' Open task window property to No .

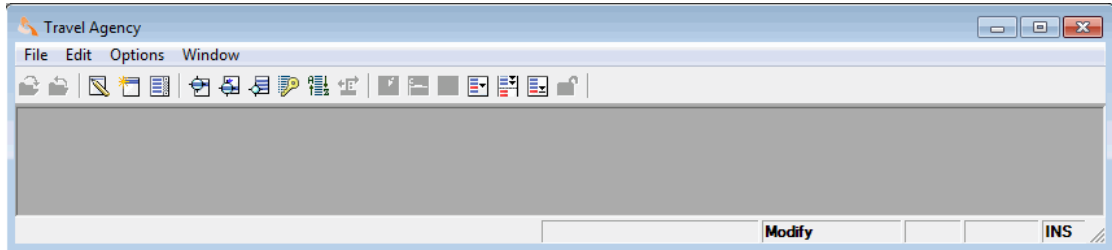
Status Bar

You now have full control over whether the status bar is displayed. By setting the **Display Status Bar** form property to **No**, you can hide the status bar from displaying. Bear in mind that status line error messages such as "Invalid Date" or "Duplicate Index" will no longer be displayed to the end user. Your own developer-defined Verify Operation messages that are defined as **Status** will also not be displayed.

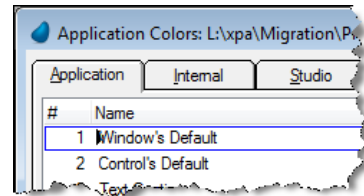
The current user name is no longer displayed on the status bar. You can address this problem by displaying the user with the **User(1)** function anywhere on your MDI form. You can check the *Magic xpa Help* for an explanation about this function.

MDI Color

In V1, the background color of the MDI was always the standard grey color as seen in the image below:

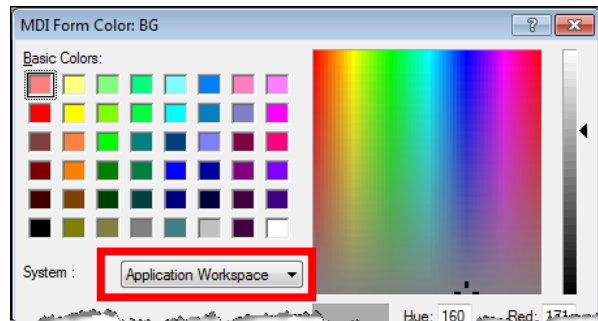


This was hard-coded with no option to change it. In Magic xpa, you have full control over the color displayed in the Main Program form. The MDI area uses the color set in the **Color** form property. After migrating the application, the default MDI color is the default Application Color **Window's Default** color.



This is a different color from the MDI area of the previous version. For backward compatibility you can do the following:

1. Zoom into the **Application Colors** dialog box.
2. Add a new color and name it **MDI Form Color**.
3. Zoom into the background color (**BG**).
4. From the **System** property combo box, select **Application Workspace**.
5. Use the new color in the **Color** property.



In Magic xpa, the **MDI Form** color is one of the colors provided out-of-the-box in the standard `clr_rnt.eng` file.

MDI Client Area

The Main Program form defines your MDI. By setting the **Width** and **Height** properties, you have control over the size of the MDI. Bear in mind that other programs with Window Types set to **MDI Child** are displayed within this area. They cannot be larger than the MDI frame. They will be resized during runtime.

The conversion process sets the **Startup Mode** of the Main Program form to **Maximized**. However, you need to take the form size into consideration in future development projects. You should also taken into consideration that the MDI pulldown menu does not support a multi-line menu.

Exercise

Open the Chocolate application that you migrated in the previous lesson.

- Set the application caption to **Chocolate Store**.
- Set the application icon to the **msu.ico** icon. This icon can be found in the **env** directory.

No solution is provided for this exercise.

Summary

The Magic xpa MDI frame is no longer hard-coded. You have control of whether it appears, its look and feel, and what is displayed and what is not. All of this is part of the Main Program.

The Main Program's form behaves like other forms and, by default, the form is displayed. This is now your MDI frame.

You also learned what properties were moved to the Main Program.

Lesson 4

UI Changes

V3 introduces a new Visual Studio-based Form Designer for Display forms as well as a new Expression Editor.

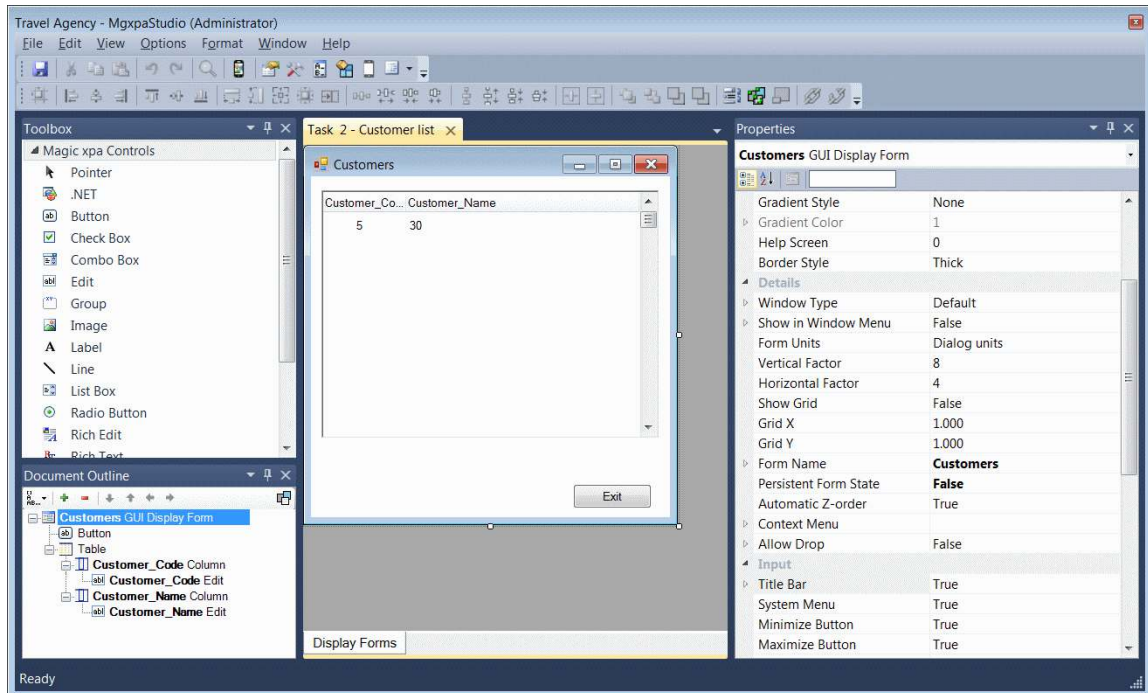
The GUI controls provided by Magic xpa and displayed on the client computer are .NET controls from the Windows Forms .NET library. As a result of this upgrade, some control properties that were available in previous versions, are no longer available and other properties have merged.

You will learn more about the changes in this lesson.

This lesson covers several topics including:

- The new Form Designer
- The new Expression Editor
- Changes to control properties
- Changes to controls
- Unsupported properties
- Unsupported controls

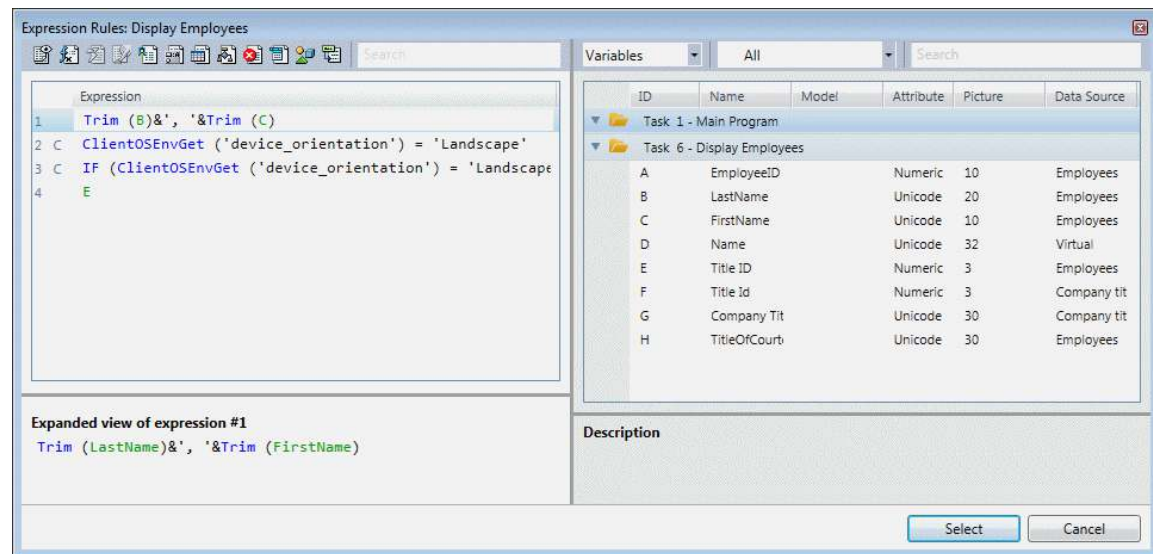
Form Designer



The Form Designer for Display forms is based on Visual Studio and includes:

- A Visual Studio-based workspace
- Docking capabilities
- Mobile Preview functionality
- A Toolbox that replaces the Control palette
- A Variable pane that replaces the Variable palette
- A Model pane
- A Document Outline view that allows you to locate controls and move them to container controls
- Quick search capabilities in the property list and pick lists that are opened from the designer
- An improved Table control with visual indication for dropping controls over it and automatic attachment of controls to the proper columns

Expression Editor



The Expression Editor was updated to include:

- Improved Auto Complete functionality for functions
- On the fly coloring of the expression and expanded view
- Improved error indications
- Embedded pick lists in the right pane
- Quick search capabilities

Form Properties

There are some changes to the form properties.

In the **Wallpaper** properties, Truevision targa images (TGA) are no longer supported.

The **Automatic Tab Order** and **Automatic Z-order** options were moved from the Command palette to the form properties.

Startup Position

The **Startup Position** of **Default** was renamed to **OS Default Bounds**. In this option, the form is positioned according to the operating system's default location and the initial size is determined by the operating system.

A new option was added named **OS Default Location**. In this option, the form is positioned according to the operating system's default location but the dimensions of the form are governed by the measurements defined on the Magic xpa form by the developer.

Form State Identifier

The Form State Identifier enables the user to keep the state of the form between sessions. In V1, you provided a name for the form. The state of each form was saved individually according to the name you provided.

The **Form State Identifier** property has changed to **Persistent Form State** and the optional values are True/False. You no longer provide a name.

As in V1.x, you can use the **FormStateClear()** function to return the state back to the default, the properties set by the developer. A blank parameter, "", will clear the current form. An asterisk, '*', clears all of the current project's forms. That means that if you cleared a form named **MSU** using the function: **FormStateClear('MSU')**, you will need to manually change it to: **FormStateClear("")**.

Control Properties

Expressions are re-evaluated only if the expression contains a variable that was changed. In previous versions, the expression was re-evaluated whenever a recompute occurred.

Top / Left

To standardize both the Rich Client and the Online paradigms, the Left and the Top properties have been renamed to **X**, for Left, and **Y** for Top. The functionality remains unchanged.

The **CLeft()** and **CTop()** functions have now been renamed to the existing Rich Client functions, **CX()** and **CY()**. This facilitates a move from one paradigm to the other.

The **CLeftMDI()** and the **CTopMDI()** functions have been removed. In previous versions, these functions returned the position of a control relative to the MDI.

When the control is linked to a container, such as a Tab or a Group control, the **X** and **Y** values are relative to the container and not to the form.

This means that:



- Any expression used in the location properties of controls that are attached to a container control will not work correctly and should be changed.
- Placement values of controls that are attached to a container control may not work correctly and should be changed.

Height / Width

Online forms now have a minimum **Height** and **Width**, which is set by the Windows operating system.

The minimum size is set by **Windows.Forms** based on the **Border Style** of the form.

If you need to define smaller forms, you can use the following workaround:

1. Add the following line in the Task Prefix of the program:

```
DNSet(DotNet.System.Windows.Forms.Form.FromHandle(WinHWND(0)).Size,  
DotNet.System.Drawing.Size(65,24))
```

65 and **24** are the form's **Width** and **Height**.

2. Add the following two .NET libraries to the CRR:

- System.Drawing
- System.Windows.Forms

Style and Border Style

These properties were merged into a single property, either **Style** or **Border**. Some properties are not supported by Windows Forms and are therefore no longer supported in Magic xpa.

As an example of the change, look at the Edit control. In V1, all of the following properties were available:

Style	Border Style
2-D	Thin
3-D Raised	Thick
3-D Sunken	No Border
Windows 3D	

The ones that are no longer valid are crossed out. These are not supported in Windows Forms and have been merged into the **Border** property. Valid values are **Yes** or **No**. As an example, an Edit control on a table will be **Border=No**. This is also how you would display a 2-D Edit control.

Horizontal Alignment

The **Horizontal alignment** property for List Box, Combo Box, Tab, Group, Rich Edit controls is not supported.

Note: Right and Left alignment on some controls can be achieved using the **RTL** property.

Z-Order Changes

Windows Forms do not allow controls that are not connected to the container (in other words, not linked to the container) to be displayed “on top” of the container. As a result they will appear behind the container. They will not be visible unless the container itself is invisible.

The migration process will automatically attach such controls to the container control.

The following are some special cases regarding the container controls:

Container	Previous versions	Magic xpa
Tab	Unattached controls appeared on each tab.	The controls will now be behind the Tab control. The migration process will automatically attach these controls to the first tab.
Table	Controls such as Edit controls and Text controls that were manually added to the Column header, appeared in the table header.	The controls will now be behind the table.
Table	Controls could be stretched to more than one column	The controls will now be seen in one column only.

Text Controls

The Text or Static control has now been renamed to **Label**. The functionality remains the same.

You can now use an expression to set the text displayed in the label.

The **Line Width** and **Line Style** properties are no longer supported.

RTF-related properties were removed from the Label control and Label-related properties were removed from the RTF control. To achieve the same functionality as Enable RTF=Yes, you can define an RTF control.

RTF Controls

The RTF Edit control has no built in context menu. If you want to give the end user the ability to change the color or font of an RTF control, you can define a menu with the Magic xpa internal events, such as Change Color, Change Font, Align Left, and Align Right, and assign this menu to the control's **Context Menu** property.

The RTF Edit control and RTF Text control no longer support a transparent background color.

Edit Controls

As in V1, if an Edit control is not large enough to display all the data, the user is able to click the **Wide** keyboard shortcut (F6) in order to increase the display; thereby displaying more data. In V1, you were able to define a form that would be displayed when the user selected the **Wide** operation (F6). As of Magic xpa, the developer defined form is no longer supported. The default Magic xpa expansion window is displayed instead. You can reproduce the previous behavior programmatically by trapping the Wide event and calling your own program. If you used a form in a previous version for the Expansion Window option, it is not removed by the conversion utility. You can remove it manually.

In an Edit control with Multi-line edit, you were able to define whether to display the scroll bars. The **Show Scroll bars** property has been removed. The **Word Wrap** option has been added to the **Horizontal scroll** property.

When parking on an Edit control and the value of the variable attached to the control is changed by an Update operation, the new value will be seen immediately (in V1 it was seen after leaving the control).

Numeric calculations (such as writing '2+' in the control) are no longer supported.

Radio Button Controls

The **Multiline** property for Radio Button controls is not supported when the **Appearance** property is set to **Button**.

Static Controls

The following properties are not supported in Static controls:

- Static Type
- Line Width
- Line Style

Table Controls

In V1 it was possible to define a specific column as a marking column. To do this you opened the Column properties sheet and defined Marking column=Yes. In Magic xpa, a row is marked by clicking on it. The **Marking column** property has been removed.

The following properties are also no longer supported:

- Style
- Table in Window
- Last Divider
- Keep Width

It is not possible to use a List Box control in a Table control.

Double clicking on the table column divider will not resize the column.

Scrolling with the mouse is only enabled when parking in the table.

Image Controls

The Image control is now a display-only control. All file input capabilities have been removed. All input properties, such as **Must Input**, have been removed.

The **Border** options are now **2D**, **3D Sunken** or **No Border**. The image default (inherited value) is now **No Border**.

Truevision targa (.TGA) images are no longer supported.

Push Button Controls

The following changes have been made to the Push Button control:

- The name of the control was changed from a Push Button control to a Button control.
- The **Text on Image** Button style property and the **Image Button** Button style property have been merged. The style is now **Image Button**. The conversion utility will convert buttons that were set with the **Text on Image** option into Image buttons. To reproduce the **Text on Image** feature, add text to the **Format** property.
- The Hyper Text button color is now taken from the Windows color definitions.

Choice Controls

Radio buttons, list boxes and combo boxes have their various options displayed from three properties: **Items List**, **Display List** and **Data Source**. All these options have now been grouped under a new group named **Data Source**. The **Source table** property has been renamed to **Data source number**.

Data Source	
Items List	Female, Male
Display List	
Data source number	0
Display Field	0
Linked Field	0
Index	0
Field Ranges	0

Group Controls

The location of the groups' top line is set according to the Group control's font, even if no text is set for the control. If you want the line to appear at the top of the Group control, you need to use a font with a size of 1.

Tab Controls

In V1, you needed to provide a default value for the Tab control so that it would park on the first tab. In Magic xpa, the first tab is always selected by default.

The **Right** and **Left** options of the **Tab Control Side** property were removed.

The Tab control header color is defined by the Windows operating system and cannot be manually set.

Tree Controls

The Tree control does not support multi-marking, drag and drop, or a transparent background color.

Subform Controls

In V1, the Subform behavior was hard-coded. When the parent task loaded, so did the Subform task. When the Subform was either connected to a container, such as a Tab control, or else had an expression on its visibility, there was difficulty in understanding when the Subform task would be loaded.

Magic xpa provides you with a new property, **Refresh When Hidden**, so that you can define when the Subform loads. You also have the ability to load and refresh the task running in the Subform control in the same way as it was done in previous versions.

A value of **No** indicates that the Subform task will only be called if the Subform is visible. This is similar to the behavior in previous versions when the Subform had an expression on the **Visible** property.

A value of **Yes** means that the Subform tasks will be called initially after the Record Prefix of the host and again after every refresh of the Subform. This is similar to the behavior in previous versions when the Subform was attached to a container control and the container control was hidden, such as a specific tab in the Tab control.

If you have nested Subforms defined with a value of **Refresh When Hidden=No**, the nested Subform will behave as the parent. Therefore if the parent's Subform is defined with a value of **Refresh When Hidden=Yes**, then when the parent refreshes, the hidden nested Subform will be refreshed even though the child's Subform is defined with **Refresh When Hidden=No**.

In V1, when the user clicked on a Subform control, the Control Prefix **and** Control Suffix of the first control were always executed. In Magic xpa, the behavior has now changed. Only the Control Prefix of the **clicked** control will be executed.

If an Online task running in a Subform control cannot be executed, the subform will close (and not the entire program, as in V1).

Unsupported Controls

Some controls were removed because they are not supported in .NET:

Control	Comment
Splitter	The Splitter control is no longer available. The functionality is achieved using the new GUI frame feature. You will learn about GUI frames in the next lesson.
Horizontal Slider and Vertical Slider	Not supported. Use a third-party .NET control. If you defined Slider controls as models, they will still appear in the project's Model repository after the conversion. You will need to remove these models. The conversion process will automatically remove Slider controls defined in a form. However, any variable or expressions used in the Slider controls' properties will still exist and may result in a warning message when you run the Syntax Checker.
Ellipse	Not supported. Use a third-party .NET control.
Rectangle	Not supported. The conversion utility converts this to a Label control.

ActiveX and OLE Controls

ActiveX and OLE controls are not supported in the Windows Forms framework. The conversion process does not remove the ActiveX and OLE controls. They are kept for information only. The program will not pass the Syntax Checker.

You need to either:

- Use the Microsoft **aximp** .exe utility, which can create the .NET class wrapper for you. This has an advantage in that you already know the exposed functions. However, this method is not recommended as it will not always work properly.
- Use native or third party .NET controls to replace the ActiveX and OLE controls. If you defined ActiveX or OLE controls as models, they will still appear in the project's Model repository after the conversion. You will need to remove these models.

Non-visual COM objects that are not ActiveX are still supported.

Summary

The change to Windows Forms resulted in welcome changes to many of the Magic xpa controls. However, some properties that you were used to, are no longer supported in the new version.

Controls such as the Ellipse, the Rectangle and the Sliders are not supported. If you need these controls, you can use .NET controls to achieve the same functionality.

The ActiveX control is no longer supported. However, you can wrap the control with a .NET class.

The Splitter form was upgraded and is now a GUI Frame. You will learn about GUI frames in the next lesson.

Lesson 5

GUI Frames

Often you need to create more sophisticated forms. Frames enable you to divide a task into multiple sections, each of which displays a different program.

Frames work similarly to Subforms. However, with a frame, the user can easily resize each section by dragging the divider bar.

Frames can be nested, and also used in conjunction with Subforms and tabs, to create elaborate user interfaces.

This lesson covers several topics including:

- Using frames
- Frame name
- Dynamic display

Why Use Frames?

The best feature of frames is that the end user can define how much information to display by resizing each section. If you have an order-lines scenario and you use a Subform, the display remains as defined by you, the developer. However, by using a frame, users can then decide that they want to increase the lines section to show more lines. The decision is in the users' hands.

With a frame you can keep one part of the page static while changing another part. So you can use it for navigation menus, similar to the Windows Explorer view.

Frames allow you to keep pertinent information always on the screen. One frame is always in view so that you can use this as the master frame. You can also show other information that has no relation to the master frame.

Using Frames

The **GUI Frames** form is an **Interface Type** of **Class 0**. A frame can only be used in interactive programs.

In this example you will have a simple scenario in which you will display the list of customers and for each customer show the list of trips they have taken, a simple one-to-many scenario.

1. From the **Frames** folder, copy the **Customers** program. This program displays the list of customers as well as a few personal details about each customer.
2. Zoom into the Form Editor.



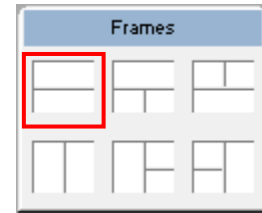
When you execute a Magic xpa program, the first form of the program with **Class=0** is the default form, the form that will be displayed in runtime. In this example you want to display a frame. However, you already have a pre-designed form. You can change the **Interface Type** from **GUI Display** to **GUI Frames**, but that will remove all the controls on the form. To handle this issue, you can duplicate the form and define the first form as a GUI Frame.


3. Park on the **Customers** form and select **Repeat Entry** from the **Edit > Entries** pulldown menu. You can also use the **Ctrl+Shift+R** keyboard shortcut. Duplicate the entry.
4. Park on the first **Customers** entry and rename it to **Customers Frame**. This is simply to distinguish it from the regular form. Set the **Interface Type** to **GUI Frames**. You will get a Confirmation dialog box: **All controls will be deleted**. Select **Yes**.

#	Name	Class	Area	Interface Type
1	Main Program		0	GUI Display
2	Customers Frame		0	GUI Frames
3	Customers		0	GUI Display

The first step in a frame set design is to select the main frame structure. Magic xpa provides you with pre-defined frames so that you can divide your frame horizontally or vertically. Each frame is a different program. In this example, you will be dividing the frame horizontally.

- Zoom into the GUI Frames form named **Customers Frame**. You will see the palette where you select the format of your frames. Select the top left option, which divides the frame horizontally.



 After defining a GUI Frame and setting up how the frame is divided up, you are unable to change the divisions. If changes need to be made, you will need to redefine the frame divisions.

Each frame displays a different program. In this case, the upper frame will be the master frame. Here you want to display the current program, the customer list.


- Park on the upper frame and zoom into the frame properties.

The **Connect to** property enables you to select whether a program, a subtask or a form will be displayed in the frame area.

- Set the **Connect to** property to **Form**.
- Zoom from the **PRG/TSK/FRM num** property and select the **Customers** form.

The **Customers** form will now be displayed in the upper frame. However, you want the dimensions of the frame to fit the current form size.

- Park on the frame in the Form Editor and open the context menu. Select **Fit to form**. This will set the dimensions of this frame to meet the dimensions of the form.

 When you define the **Connect to** property as **Form**, you are setting the primary frame for the program.

You now want to display the **Orders** program in the lower frame.

10. Park on the lower frame and zoom into the Frame properties.
11. Set the **Connect to** property to **Program**.
12. Zoom from the **PRG/TSK/FRM num** property and select the **Orders** program.
13. The **Orders** program requires a parameter. Zoom into the **Arguments** dialog box.
14. Create a line and select the **Customer_code** variable.

Now when you run the **Customers** program you will have a frame. You can increase the size of the upper frame or the lower frame by moving the dividing bar.



When the GUI Frame is defined as the first form, it is automatically displayed when the program is executed. How can you define that a different form will be displayed instead?

The GUI Frame is a form like any other form and therefore may be used in an expression using the **FORM** literal. You can therefore define an expression in the **Main Display** task property. As an example, you can have an expression such as: **IF (Logical expression, 2'FORM',3'FORM')**. It is good practice to use the literal to avoid problems if you move the forms around. Remember that this expression is evaluated when the program loads.

Automatic Refresh

In the example, you saw a situation in which there is a one-to-many scenario, the bottom pane behaving like a Subform. The frame property, **Automatic Refresh**, behaves as it does in a Subform, meaning that when set to **Yes**, the frame view is automatically refreshed when any of the parameters are changed, in this case the **Customer_code**. This property is not available when the **Connect to** property is set to **Form**. When it is set to **Form** you are referring to the current program, so there are no parameters passed that affect the behavior of the frame.

When a frame is defined with **Automatic Refresh** set to **Yes**, then whenever any of the arguments passed to the program are changed, this will invoke a change in the called program. If the called program has a range based on the parameter, then new records will be fetched from the database. This has an effect on the performance of the system. For a large data set, this may cause unwanted overhead.

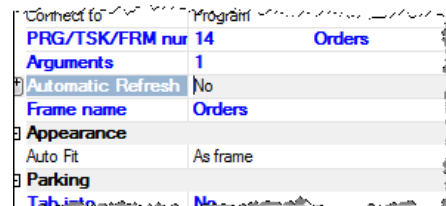
When you pass a number of parameters in which each parameter may be updated in the parent record, you may decide to set **Automatic Refresh** to **No**. This will cause less overhead. An example of a form like this would be a Search filter in which the user

would enter the search parameters and click an action button. In the logic unit of the event invoked by the button you would manually refresh the frame.

Frame Name

In the previous hint it was recommended to set **Automatic Refresh** to **No** for cases where there are many arguments that may be modified. In those cases, it is good practice to use a push button. But, how do you perform a refresh of the frame?

One of the properties of a frame is the **Frame name**. This name must be unique in order to identify it.



Connected to	Program
PRG/TSK/FRM nur 14	Orders
Arguments	1
Automatic Refresh	No
Frame name	Orders
Appearance	
Auto Fit	As frame
Parking	
Tab into	No

The **Subform Refresh** Internal event receives a **Frame name** as a parameter and refreshes the frame indicated by the name. In this way you can manually refresh the frame. The necessary steps would be:

- Define an event, **Refresh Orders Frame**.
- Create a logic unit for the **Refresh Orders Frame** event.
- Raise the **Subform Refresh** event and pass the name of the frame you want to refresh, such as **Orders**.

Dynamic Display

The dimensions of the frame are as defined by the developer. The user can decide to decrease or increase a section by moving the divider. But how can you dynamically remove a frame? You are probably asking: Why would I want to? You may have a situation where the form is split vertically and the right frame split horizontally, similar to the fifth option in the Frames palette. In the left frame you display a menu and according to whatever is clicked, you want the right pane to display one or two frames.

When you display a frame, you call a different program or form. This form may also be a frame so that you have a situation with nested frames. One solution for nested frames is to have the called program define whether it is a form or a nested frame. When the secondary frame has a relationship such as a one-to-many scenario, this is a reasonable solution. However, what happens when the programs displayed in the frame are not dependent on one another, such as an overall view or a dashboard view?

Another solution is to have an expression on the dimensions of the secondary frame, either the height or the width, depending on whether the frame is split horizontally or vertically. You can then define the size as zero. When you define the size as zero, the divider is still visible and the program is still called. When the size is evaluated to zero, you will still need to define a program that will be displayed. However, this can

be a dummy program. You will also need to set the **Tab Into** property and the **Enable Park** property to **No**.



The Call Program operation has a **Destination** property. Here you set the name of the frame where you want the program to be displayed.

This is also useful if you want to replace the program in a Subform.

Retain Focus

The Call operation has a property called **Retain Focus**. When you call a destination Subform or Frame, this property defines whether focus remains in the current task and control or is moved to the first control of the called program. It is enabled only when:

- The **Destination** property has a value.
- The Call Program operation is called from a logic unit other than the Task, Record or Control logic units.
- The program is not the Main Program.

Splitter Conversion

The splitter from V1 is no longer supported. When you defined a splitter you called a program from the Record Prefix logic unit that would be displayed in the splitter section.

This program generally had the following characteristics:

- An End task condition set to: Level (1) = 'RP'
- A Window Type set to Splitter Child

The conversion process for a program containing a splitter does the following:

- Adds a **GUI Frames** form as the main form of the program. The added form will include a reference to the existing form in one of the frames.
- Converts the Splitter window into a GUI Frame with the same dimensions. The V1.x splitter window could only be split horizontally or vertically, so the GUI Frame is set accordingly.
- Evaluates which frame was defined as the V1.x **Primary Display** property and replaces that by setting the **Connect To** property to **Form** and displays the original form.

After the application is converted, you need to execute a number of steps manually. You may already have worked out what steps you need to perform. You will need to make changes both in the GUI Frame program and the called program.

In previous versions, in the splitter program, you called a program from the Record Prefix. This program would be displayed in the splitter pane, if the **Window Type** was set to **Splitter Child**. In Magic xpa, you need to make the following changes:

1. Zoom into the splitter converted program. As an example, the **Customers** program from the Conversion folder.
2. Zoom into the **GUI Frames** form and park on the secondary frame, the one that is currently not connected to a program, task or form.
3. Set the **Connect to** property to **Program**.
4. Set the **PRG/TSK/FRM num** to point to the **Orders** program.
5. Zoom into the **Arguments** property and send the **Customer_Code** variable to the **Orders** program.

You now need to remove the Call Program operation itself.

6. Zoom into the **Record Prefix** logic unit.
7. Delete the **Call Program** operation to the **Orders** program.



If you called either program A or program B depending on a certain condition, you will need to implement the same logic in the frame itself.

However, if you called program A or subtask B depending on a condition, you will need to implement subtask B as a program. The GUI Frame does not currently enable you to have an expression for the **Connect to** property.

In the called program, you only need to remove the end task condition:

8. Zoom into the called program such as the **Orders** program from the Conversion folder.
9. Open the **Task Properties** dialog box and set the **End Task Condition** property to **No**. Since the program is no longer called from the Record Prefix, there is no meaning to the original condition. You can also set the **Evaluate condition** property to **Before entering record**.
10. The form property, **Window Type** was set to **Child** by the conversion program. Splitter Child is no longer supported. You do not need to make any changes here.

11. The **SplitterOffset** function was removed. The Conversion utility will not delete the removed function from the Expression Editor. You should manually fix the syntax errors caused by this change.
12. A model of a splitter form is converted into two models – one for the frameset and one for the display form. If the form has a model attached from a component, you need to manually assign the display form model to the display form.

Splitter Placement

The **Splitter placement** property is no longer supported. It is converted automatically to either **Horizontal Placement** or **Vertical Placement**. The conversion utility makes the following conversion: The new placement options are either **Yes** or **No**.

- If the Splitter placement value was 100, then the master frame will have **Vertical Placement=Yes** and the secondary frame will have **Vertical Placement=No**.
- If the Splitter placement value was 0, then the master frame will have **Vertical Placement=No** and the secondary frame will have **Vertical Placement=Yes**.
- Another value will have both placement values set to **Yes**.

Exercises

1. Define a program named **Possible Locations**. Display the list of locations that are available as travel destinations. This program must also display the list of airlines that fly to those locations as well as the list of tourists who flew to those cities.
Implement this with frames so that the users can change the dimensions of the frames themselves.
2. As an extra example, in the top frame of the **Possible Locations** program, also show the list of deals, past and present to that location. Give thought to the solution here.
3. During the lesson, you defined a program where you displayed the list of customers and the list of their orders. Each order also has details, the locations and the dates of each trip. Add the details of each order in a frame to the right of the list of orders.

Summary

In this lesson, you learned about the GUI Frame. Frames were first implemented in the previous version in interactive programs for the Rich Client environment. It has been extended to the GUI framework to enable you to take advantage of the new functionality.

You learned how to implement the GUI Frame and learned some tips that will improve the performance of your application.

You learned what steps you need to perform when you upgrade your application from previous versions.

Lesson 6

End User Functionality

Enabling the end user to perform their own ranges, locates and sorts is an integral part of the usability of any application. Enabling the end user to define their own reports on-the-fly brings their application to new levels. This was always a part of the Magic xpa offering.

In V1, the functionality provided by Magic xpa was hard-coded into the engine.

The End User functionality is now provided as a component, providing an interface for the user to perform the Range, Locate, Sort, and Print Data capabilities.

The source code for this component is also available so that you can customize the screens of the component as you wish.

This lesson covers several topics including:

- End user component
- Range, Locate and Sort
- Print Data

Adding End User Functionality

The component is added by default to any new project. As you already learned, the conversion application enables you to add the component when migrating your application.

If you elected not to add this during the migration you can add it at a later stage:

1. In the **Composite Resource Repository (CRR)**, create a new line.
2. Zoom to select the **UserFunctionality.eci** application from the **Add_On\UserFunctionality** directory.

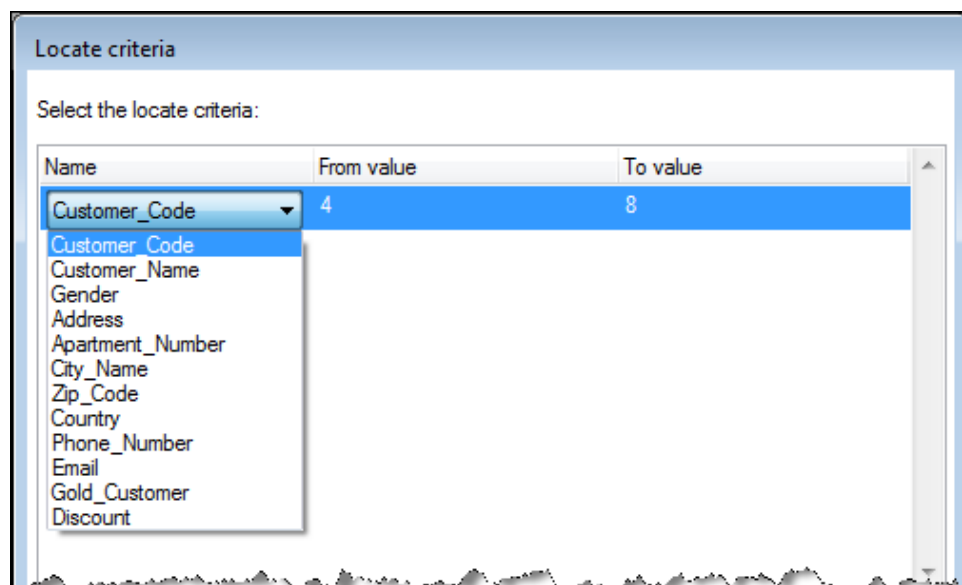


Unlike V1, if you do not add this component, the end user will not have the functionality; Range, Sort and Locate will not work.

The End User Functionality component was built using the various Magic xpa **DataView** functions. As a result, when using this component's functionality on a task that has two variables with the same name, only the first variable will be used. As in previous versions, you can rename a variable or a column in the task so that it will be visible in the dialog box.

Locate Row

Locate enables you to park on a certain row in the data view. When you invoke **Locate Records (CTRL+L)**, you are provided with a dialog box for you to select the locate criteria.



In the **Name** column, you have a combo box with all of the variable names from the task. Remember that these are displayed to the end user, so instead of having a variable named **CustCod**, you may want to provide a more meaningful name, such as **Customer code**.

You can add additional search criteria by adding a new row. You do this using the **Create Records** keyboard shortcut (F4).

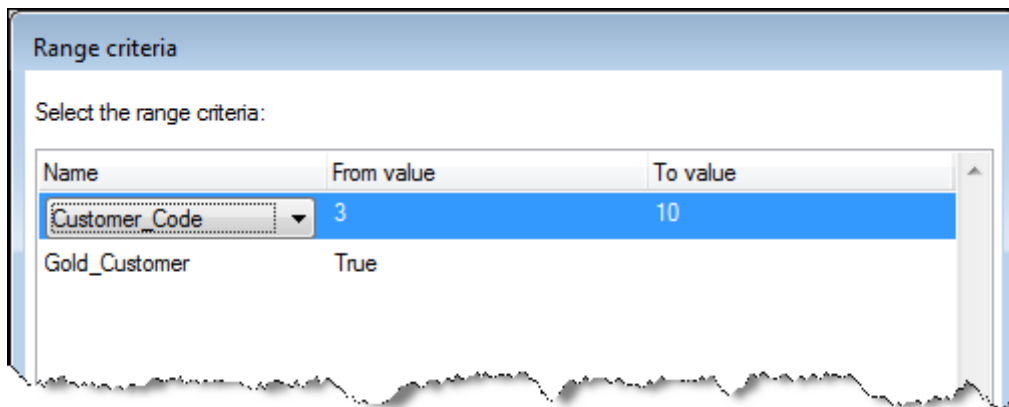


Remember that if identical names are used in a task, only the first will be used.

Expressions are not supported.

Ranges

The **Range of records** capability is now provided as a part of the End User Functionality component. When you invoke **Range of records** (CTRL+R), you are provided with a dialog box for you to select the criteria.

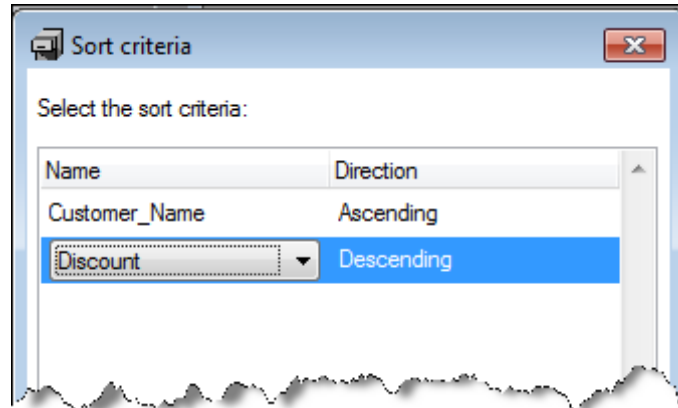


You work with this dialog box in the same way that you work with the **Locate record** dialog box.

Sorting

The Sort dialog box works very similar to the Locate and Range dialog boxes. However, here you define whether you want the selected variable to be sorted ascending or descending. You can access the dialog box using the **CTRL+T** shortcut.

You can create various levels for the same sort, the top level being at the top of the table. In the image on the right, the table will first be sorted by the customer name and then by the discount in which the discount is displayed with the highest discount first, in other words in descending order.

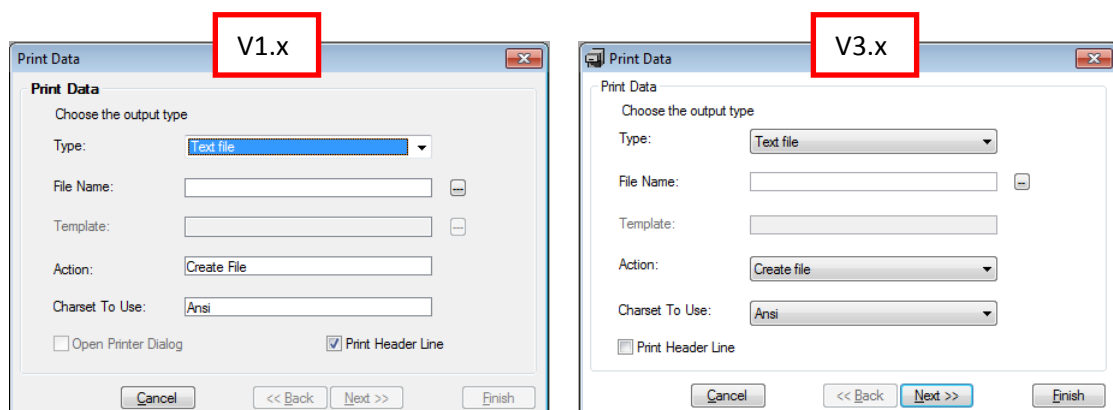


Print Data

The Print Data is now provided as part of the End User Functionality component. Remember that to enable the end user to use Print Data, you need to enable this option in the **Task Properties** dialog box. It is normally set to **No**.

The utility uses the DataViewToXXX functions that were released in previous versions.

When the end user wants to print, they receive the **Print Data** dialog box. The look and feel of the **Print Data** dialog box is similar to the V1.x dialog box.



When selecting the **Text file** option from the **Type** property, the **Create File and Print** action is not supported. Therefore, the **Open Printer Dialog** check box is also not displayed.

When selecting the **HTML file** option from the **Type** property, the **Create File and Print** action is not supported. However, if you define the file name with an HTM or HTML extension and select **Create File and Open**, the operating system will display the file in the default program, normally a browser. You can then print from there.

If you want to provide the end user with a more sophisticated report generator, you can add the **ReportGenerator** component to your application.

Unsupported Features

The **Change index** option is no longer supported.

The **View by Key** option is no longer supported. The user can add the sort that they want.

The **Redirect Files** option is no longer supported.

In the **I/O Devices** repository, the **Media** option of **Console** is no longer supported. This will generate a Checker error. You can use a Graphic Printer with **Print Preview=Yes** to display the report.

Exercise

In this exercise you will make changes to the End User Functionality application, specifically to the Print Data mechanism. Remember to back up the project and the **ecf** file before continuing.

1. In the **Print Data** dialog box, if the user selects a Text file, then add a **Create and &Print** option. The others remain the same. For the purpose of printing you can use a Notepad.exe **/p** parameter:
notepad.exe /p <filename>
This is known to work in Windows XP and Windows 7.
2. Add a filter so that certain fields will not be displayed in the **Print Data** dialog box. Implement this by adding ****** before the name of a variable.

Summary

The End User functionality provided in previous versions was part of the engine. You had no control over the display.

The End User Functionality component is provided as an open source. You can make the changes you require to improve the end-user experience. In the exercise, you made some changes. Remember that you need to back up the previous project. Take into account that when you upgrade to a new version, it will override your changes.

Lesson 7

Additional Changes

There are more changes to this exciting new product.

This lesson covers several topics including:

- Task behavior changes
- Environment changes
- Runtime changes

Task Behavior Changes

There are some changes that you need to be aware of that are not covered in other lessons in this seminar.

Task Properties

The **Foreground Window** property is not supported. In V1, it appeared only when the **SpecialOldZorder** special flag was set to Y.

The following task modes are not supported for Online tasks: Delete, Locate, Range, Key, Sort, Files, and Options (when used as initial task mode and raised by the user). When you convert your project, if you have selected one of the above modes, the **Initial mode** property will be blank. You should select one of the other modes.

Task Modes

The following task modes are no longer supported in an interactive task: Delete, Locate, Range, Key, Sort, Files and Options.

Control Verification

In previous versions, the Control Verification logic unit was invoked only when navigating between records with the mouse. In Magic xpa, Control Verification is invoked when navigating between records both with the keyboard and the mouse.

Changed Functions

The following functions were changed:

- **DirDlg** – Opened the Windows Choose Directory dialog box and returned a full path to the selected directory. The syntax of the DirDlg function is now the same as ClientDirDlg(*description, initial directory, show new folder*). This is because some arguments are not supported in .NET.
- **CtrlHWND** and **WinHWND** – In previous versions, these functions returned the handle of a control or a window for use in external dlls. These now return handles to .NET objects that you can use as references in a .NET library using the FormHandle function. They will no longer work in Win32 dlls.
- **HitZOrder** – The function returned the Z-Order number of a control. The function has now been removed. The Conversion utility will not delete the removed function from the Expression Editor. You should manually fix the syntax errors caused by this change.

- In the V1 Online version, some functions, such as **InTrans** and **ViewMod**, were always re-evaluated, even if there was no variable in the expression. This behavior was changed and the expression that contains functions is now re-evaluated only if the expression has a variable that was changed.

You can achieve the previous behavior as follows:

1. Add a Virtual variable (with Part of dataview=No) to the program.
2. Update the variable in the Record Prefix with False.
3. For ViewMod() – Add a handler on an Expression event with the following condition: ViewMod(0) AND A<>ViewMod(0) [where A is the variable]

In the handler, update the variable with ViewMod(0).

4. For InTrans() – Add a handler on the Control Modify internal event

In the handler, update the variable with InTrans().

That's it. Now the variable is updated accordingly and can be used on the form.

Parallel Execution

The **Modal** Window type now behaves as the old **Application Modal**, meaning that the current form locks all open windows in the application. The **Application Modal** Window type was removed.

Window Pulldown Menu

The Window menu will display all windows defined with **Show in Window menu = Yes**. The **More Windows** option is obsolete and unsupported. When migrating to Magic xpa, the **More Windows** menu entry is not removed and will still appear during deployment. Activating the option will be ignored. Consider manually removing this menu entry.

Environment Settings

In addition to the MDI environment settings that were removed and discussed in an earlier lesson, the following environment settings have also been removed:

- Deployment Custom Copyright (RtUserCopyright)
- Display Copyright Messages (CopyrightMessages)
- Indent Characters (IndentCharacters)
- Input Date (InputDate)
- Logo File (LogoFile)

- Pulldown Menu Close Timeout (MenuCloseTimeout) – Magic xpa now uses the Windows default.
- Image Cache Size (ImageCacheSize)
- Check Image Change Time (ImageCacheCheckTime)

Window Focusing

Window focusing was changed and the Windows operating system will now set the window focus according to the last control in focus. This means that:

- If during the execution of a report, a change is made in the GUI Display window, such as a focus change after raising a Next Field event, the Print Preview window will be seen behind the application window.
- If an external software is invoked from a batch program, the external software window will be seen behind the application window.

Magic xpa Runtime

The Magic xpa Runtime engine's built-in MDI was removed. This means that:

1. The Runtime engine (MgxpaRuntime.exe) can run only if a start application was defined. To let your end users execute several applications, they need a different shortcut for each application. In the shortcut's **Target** field you specify the location of the .ecf file.
2. The Runtime engine no longer provides environment editors. Instead, a new utility named **MgxpaSettings.exe** is available in the root folder of the application. This utility lets you modify the environment settings (such as settings, colors, and fonts) or the users rights without using the Studio.

Start Application

In V1, you were able to load the runtime executable and to enable the end user to select which application to open.

The Magic xpa Runtime executable will not load, unless a valid application file has been provided in the **magic.ini** property named **StartApplication**.

If you need to load a number of applications, you will need to define shortcuts and use INI overrides. Remember to include **/[MAGIC_ENV]StartApplication**. Remember that you can use the override in the shortcut path or by using an override file.

Your shortcut may now look like this:

```
"c:\MSU\Migration to V3\ MgxpaRuntime.exe" /[MAGIC_ENV]StartApplication=
"c:\MSU\Migration to V3\Travel.ecf"
```

More information can be found in the Mastering Magic xpa section of the Help.

Internal Events

In V1, when there was a **Control Hit** event with **Propagate=No** on a choice control, the clicked value was not selected and in V3 it is selected.

The following deprecated events were removed:

- Navigation events: Begin Form, Begin Page, End Form, End Page, More Windows
- Editing events: Insert Object, Mark Next Page, Mark Previous Page, Mark To Beginning, Mark To End
- Application events: About uniPaaS
- Task events: From Value, OK, Redirect Files, To Value, View by Key
- User Range/Locate events: Clear Template, Clear Value, Define Expression
- Settings events: Colors, Communications, DBMS, Databases, Environment, Fonts, HTML Styles, Keyboard Mapping, Languages, Logical Names, Logon, Print Attributes, Printers, Secret Names, Servers, Services, User Groups, User IDs, Visual Connection
- Miscellaneous events: Delete All, Help Topics, Paste Link, Update link

Encrypted SQLite Database Support

The SQLite database can now support encryption.

The encryption password is defined in the database properties.

It is not possible to migrate an existing non-encrypted SQLite database to an encrypted database. If you have an existing application with an SQLite database and you want to use encryption, you should define a different database with encryption and synchronize your data to this database.

Summary

In this lesson, you learned about other changes in the new version.

You learned about features that were modified in the program, such as the Task options, the Control Verification logic unit and functions that were modified or removed.

You learned about properties that were removed from the Magic xpa Environment interface.

You learned that the Magic xpa runtime can only be executed if a valid application file has been provided.




Solutions

Lesson 4 – GUI Frames

Travel Destinations

You are asked to define a program that displays the list of locations that are available as travel destinations.

The programs that you can use are found in the **Frames** folder of the Program repository. If you make changes to the programs, it is good practice to copy them so that you can always return to the original state.

1. Copy the program named **Locations**. Rename it to **Possible Locations**.
2. Zoom into the **Possible Locations** program.
3. Zoom into the Form Editor (**CTRL+3**).
4. Duplicate the **Possible Locations** form.
5. Set the name of the first copy to **Travel Destinations** and set the **Interface type** to **GUI Frames**. You will get a dialog box asking whether to delete all controls. Select **Yes**.
6. Zoom into the **Frames** form. Select the second option from the frames palette, which looks like the image on the right. 

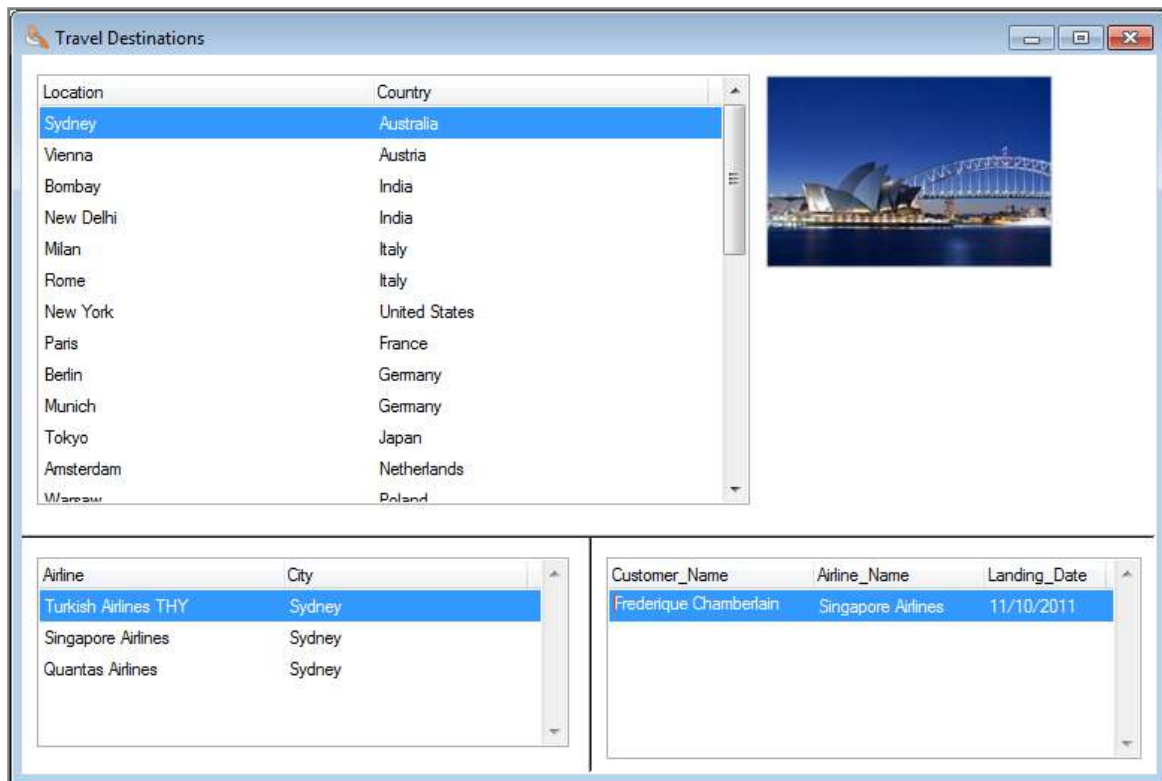
Now you will define the frames:

7. Park on the top frame. This will be the master frame.
8. In the frame properties, set the **Connect to** property to **Form**.
9. In the **PRG/TSK/FRM num** property, zoom and select the **Possible Locations** form.
10. Park on the top frame and use the context menu to select **Fit to Form**. The top frame will now be resized accordingly.

The master frame is now defined, now you need to define what will be displayed in the frame:

11. Park on the left frame in the lower half.
12. In the frame properties, set the **Connect to** property to **Program**.
13. In the **PRG/TSK/FRM num** property, zoom and select the **Airlines Flying** program.
14. The program receives a parameter. Zoom into the **Arguments** property and send the **Location_code** as an argument.
15. Park on the right frame in the lower half.
16. In the frame properties, set the **Connect to** property to **Program**.
17. In the **PRG/TSK/FRM num** property, zoom and select the **Who arrived** program.

18. The program receives a parameter. Zoom into the **Arguments** property and send the **Location_code** as an argument.
19. You can now execute the program.



List of Deals


As an extra example, you are asked to display the list of deals in the top frame of the **Possible Locations** program. In this case, using frames is not an ideal solution. Using a subform is a better solution.

1. Zoom into the **Possible Locations** program.
2. Zoom into the **Possible Locations** form.
3. Add a Subform control under the image. You need to increase the size of the form to add the Subform control.
4. Set the **Placement** to **50,0,100,100**.
5. Set the **Connect to** property to **Program**.
6. In the **PRG/TSK/FRM num** property, zoom and select the **Deals** program.
7. The program receives a parameter. Zoom into the **Arguments** property and send the **Location_code** as an argument.
8. You can now execute the program.

You have just seen an example where it was easier to use a subform rather than a frame.

Customer Orders

In the lesson, you defined a program named **Customers** that displays the list of customers and the list of their orders. However, each order also has details, the locations and the dates of each trip. You were asked to add the details of each order in a frame to the right of the list of orders. You could implement this as a subform, but then the user will have no control over its size. You will define this as a frame.

1. Zoom into the **Orders** program.
2. Zoom into the Form Editor (CTRL+3).
3. Duplicate the **Orders** form.
4. Set the name of the first copy to **Order Frames** and set the **Interface type** to **GUI Frames**. You will get a dialog box asking whether to delete all controls. Select **Yes**.
5. Zoom into the **Order Frames** form. Select the fourth option from the frames palette. 

Now you connect the original form to the left frame:

6. Park on the left frame. This will be the master frame.
7. In the frame properties, set the **Connect to** property to **Form**.
8. In the **PRG/TSK/FRM num** property, zoom and select the **Orders** form.
9. Park on the left frame and use the context menu to select **Fit to Form**. The frame will be resized accordingly.

Now you call the order details program from the right frame:

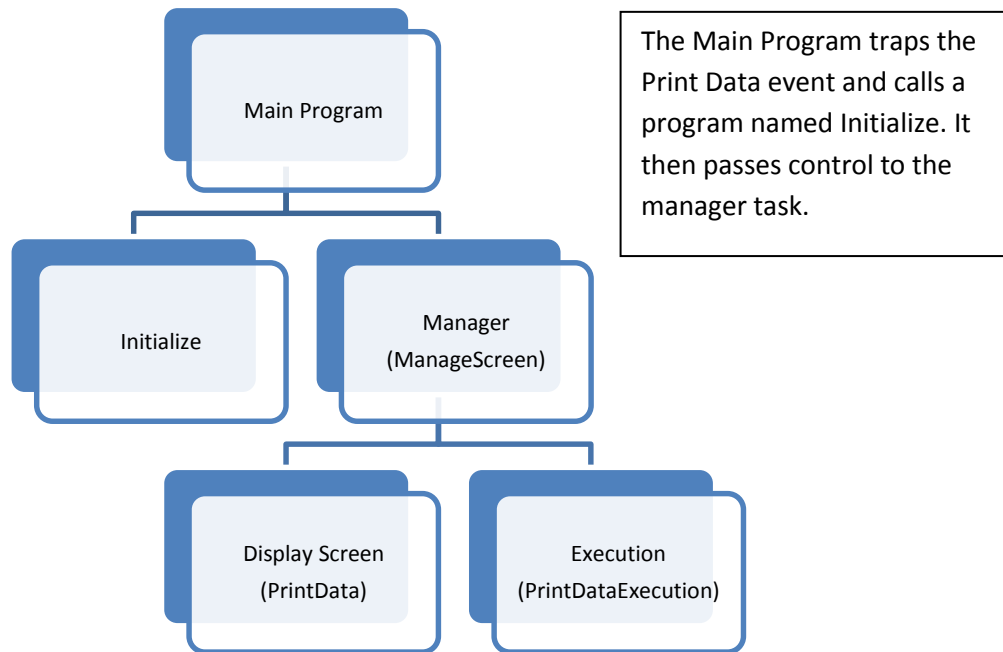
10. Park on the right frame.
11. In the frame properties, set the **Connect to** property to **Program**.
12. In the **PRG/TSK/FRM num** property, zoom and select the **Order Lines** program.
13. The program receives a parameter. Zoom into the **Arguments** property and send the **Order_Number** as an argument.
14. Park on the right frame and use the context menu to select **Fit to Form**. The frame will be resized accordingly.
15. You can now execute the program.

This solution shows an example of using a nested frame.

Lesson 5 – End User Functionality

Changing the Print Data

The Print Data mechanism in the End User functionality works like this:

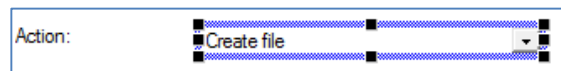


Adding Print Functionality

To add print functionality, you need to modify the display and the execution.

Before you start, back up the End User Functionality project and the ECF. The sources are found under the Magic xpa installation, in the **Add_On** directory.

1. Open the **UserFunctionality** project.
2. Zoom into the **OL:PrintData** program.
3. Zoom into the Form Designer.
4. Park on the **v.Action** combo box.



You need to make some changes here so that when the type (**v.Type**) is a Text file, the options in the combo box will change.

5. Zoom into the control properties.
6. Park on the **Items List** property. Expand the property and zoom into the Expression Editor. Enter the following expression: `'C,O'&IF (v.Type='T','P', '')`
v.Type contains whether this is Text, HTML or XML. In the expression, you are adding the **P** option if this is a text file.

7. Park on the **Display List** property and zoom into the Expression Editor. Enter the following expression:

```
'&Create file,Create and &Open file'&IF (v:Type ='T',',Create and &Print file',')
```

In the expression, you are adding the **Create and &Print file** option if this is a text file. Do not forget the preceding comma.

You now need to handle what happens if the user selects **Print**. Zoom into the **OL:PrintDataExecution** program.

8. In the **Task Suffix** logic unit, zoom into the **Block If** section.
9. Add an Invoke OS Cmd operation and set the expression to:
`'Notepad.exe /p "'&Trim(gv:FileName)&'''`
10. Set the condition for this line to `gv:Action='P'`

The `/p` parameter of Notepad will print the file and then close itself.

Remember this solution is an example and currently works on Windows XP and Windows 7.

Filtering Field Names

One of the challenges is to filter some fields that you do not want to be in the list. This is a simple process. The field names are fetched using the **DataViewVars** function. To filter a certain name you can add a prefix.

1. Open the **UserFunctionality** project.
2. Zoom into the **Initialize** program.
3. Zoom into the Data View Editor.
4. Add a Virtual Numeric variable named `v:Order` with a length of `5`. You can use the **Order** model.
5. Zoom into the Logic Editor and zoom into the **Block While** section.
6. Add a **Block If** section within the **Block While** section.
7. Set the condition to:
`Left (VecGet (v:TaskVariableList, LoopCounter ()),2)<>'**'`
If the first two characters of the variable name are `**`, then the processing will not enter the block.
8. Add an **Update** operation and update the `v:Order` variable with `v:Order+1`.
9. Move the **Call Program** operation, which is calling the **AddToVariableList** program, to within the **Block If** section as the second operation.
10. Zoom into the **Arguments** dialog box and pass the `v:Order` variable to the `pl:Idx` parameter and also to the `pl:Available` parameter.

The **Block While** section will look something like this:

Block	While	6	{LoopCounter()<=VecSize(v:TaskVariableList)
Block	If	12	{Left (VecGet (v:TaskVariableList, LoopCounter ()),2)<>""
Update	Variable	BF	v:Order With: 13 v:Order+1
Call	Program	7	AddToVariableList [3 Arguments]
Block	End		}
Block	End		}

You can now run a program for printing. Remember that if you want to eliminate a field from being displayed in the list, you need to add ** to the field name in the task.

