LABELSTAR OFFICE

Reference Manual

Version 6.50 Build 1010
March 2019

Contents

Getting Started

Labelstar Office is equipped with an Automation interface (previously also referred to as OLE Automation). The application programming interface (API) allows external applications to embed **Labelstar Office's** functionality into their own program. It enables access to the labels created with **Labelstar Office**.

Note

The sample codes are written in C# and VB.Net. Although you can use any programming language that supports Automation, such as Visual Basic or C++.

To use Automation, you must have installed a copy of **Labelstar Office** on your system.

Using the Application Object

An Application object represents the **Labelstar Office** application. This is the top level object from which all other objects will originate. Its members usually apply to **Labelstar Office** as a whole. You can use its properties and methods to control the **Labelstar Office** environment.

You must make sure that its lifetime exceeds the lifetimes of all other objects because all the other objects belong to the Application object. This means that you almost always delcare the Application object at project global scope.

There is no reason for any other project to use more than one Application object because any number of other objects can share a single Application object.

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
}
```

Example (VB.Net)

```
Dim objApp As LSOffice.Application

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize()

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
End If
```


Using the Label Object

The Application object can be used to open one or more labels by calling the Application.OpenLabel method. The returned Label object represents the **Labelstar Office** label (*.lbex) just opened and gives access to a wide range of **Labelstar Office** functions.

The Label object allows you to save and print the label, as well as modify text, bar code and image data. When using the Label object, developers can manipulate existing labels by changing print settings and field properties. It does not allow you to modify the layout or to create a new label.

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

// Open label
LSOffice.Label objLabel = objApp.OpenLabel ("C:\\Label1.lbex");

if (objLabel == null)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

// Change field content
LSOffice.Field objField = objLabel.GetFieldByName ("Text1");

if (objField == null)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

objField.SetContent ("12345");

// Print label
objLabel.Copies = 5;
objLabel.Print ();
```


Example (VB.Net)

```
Dim objApp As LSOffice.Application
Dim objLabel As LSOffice.Label
Dim objField As LSOffice.Field

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize()

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

'Open label
objLabel = objApp.OpenLabel("C:\\Label1.lbex")

If (objLabel Is Nothing) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

'Change field content
objField = objLabel.GetFieldByName("Text1")

If (objField Is Nothing) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

objField.SetContent("12345")

'Print label
objLabel.Copies = 5
objLabel.Print()
```


Using the Printer Object

The Printer object can be used to manage one or more printers by calling the Application.GetPrinter method. The returned Printer object represents a printer installed on your computer.

The Printer object allows you to check the printer status, as well as to send raw data directly to the printer. If the printer is a valid Carl Valentin label printer special print commands can be sent to the printer, e.g. to stop, continue or cancel a print job.

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

// Get printer
using (LSOffice.Printer objPrinter = objApp.GetPrinter ("LabelPrt1"))
{
   if (objPrinter == null)
   {
      MessageBox.Show (objApp.LastError.Message, "Error");
      return;
   }

   // Check printer status
   string message;
   LSOffice.PrinterStatus status = objPrinter.GetStatus (out message);

   if (status != LSOffice.PrinterStatus.Idle)
   {
      MessageBox.Show (message, "Printer Status");
      return;
   }

   // Does the selected printer supports print commands?
   if (!objPrinter.IsCVPrinter)
   {
      MessageBox.Show ("Invalid label printer.", "Error");
      return;
   }

   // Label feed
   objPrinter.Execute (LSOffice.PrintCommand.LabelFeed);

   // Test print
   objPrinter.Execute ("(SOH)FF----r(ETB)");
}
```


Example (VB.Net)

```
Dim objApp As LSOffice.Application
Dim objPrinter As LSOffice.Printer

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize()

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

'Get printer
objPrinter = objApp.GetPrinter("LabelPrt1")

If (objPrinter Is Nothing) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

'Check printer status
Dim message As String = ""
Dim status = objPrinter.GetStatus(message)

If (status <> LSOffice.PrinterStatus.Idle) Then
   MsgBox(message, vbOKOnly, "Printer Status")
   objPrinter.Dispose()
   Return
End If

'Does the selected printer supports print commands?
If (Not objPrinter.IsCVPrinter) Then
   MsgBox("Invalid label printer.", vbOKOnly, "Error")
   objPrinter.Dispose()
   Return
End If

'Label feed
objPrinter.Execute(LSOffice.PrintCommand.LabelFeed)

'Test print
objPrinter.Execute("(SOH)FF----r(ETB)")
objPrinter.Dispose()
```


Creating Your First Application

The best way to learn how to create an application with Automation is to actually create one. In this section you will create a simple application to open and print labels.

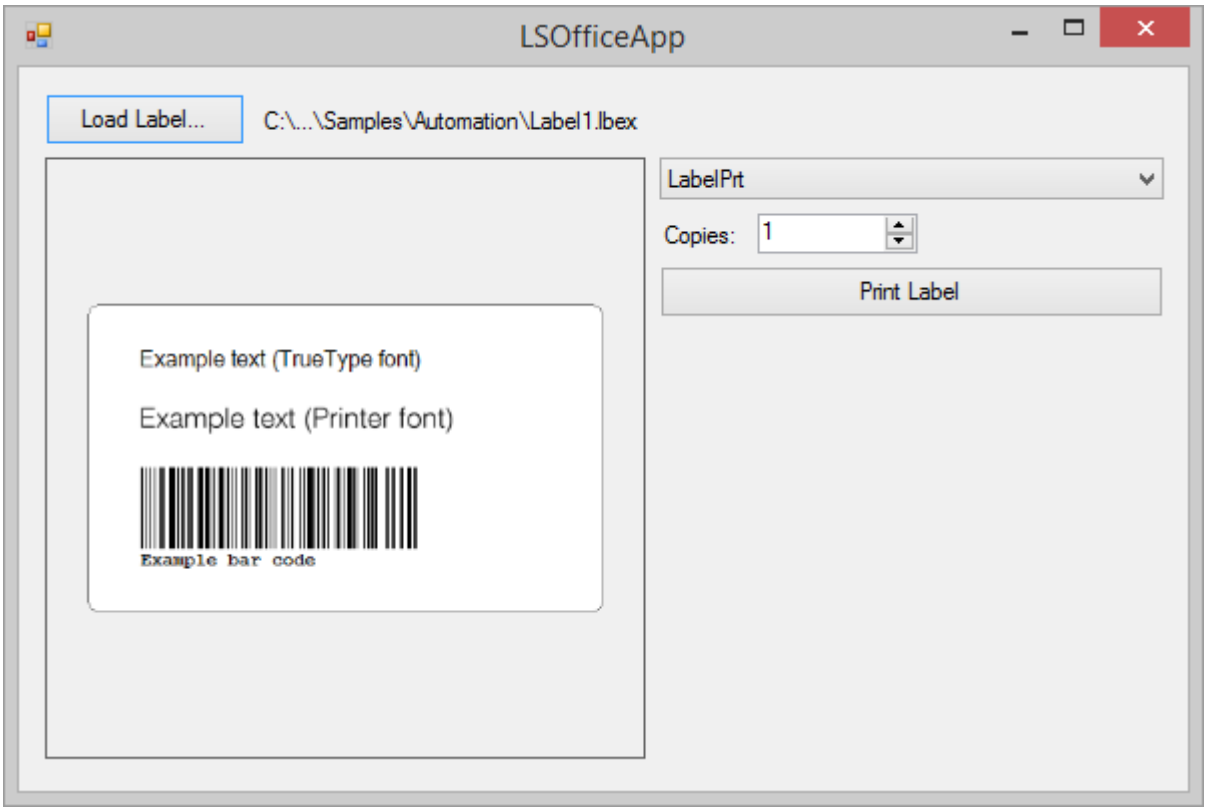
Note

The application is created using Microsoft Visual Studio 2015 and a copy of **Labelstar Office** must be installed on your system.

You learn how to:

- Create a new project
- Include the LSOffice API in your project
- Initialize the Application object
- Open and print a label

When you finish, your program will look like the following picture.



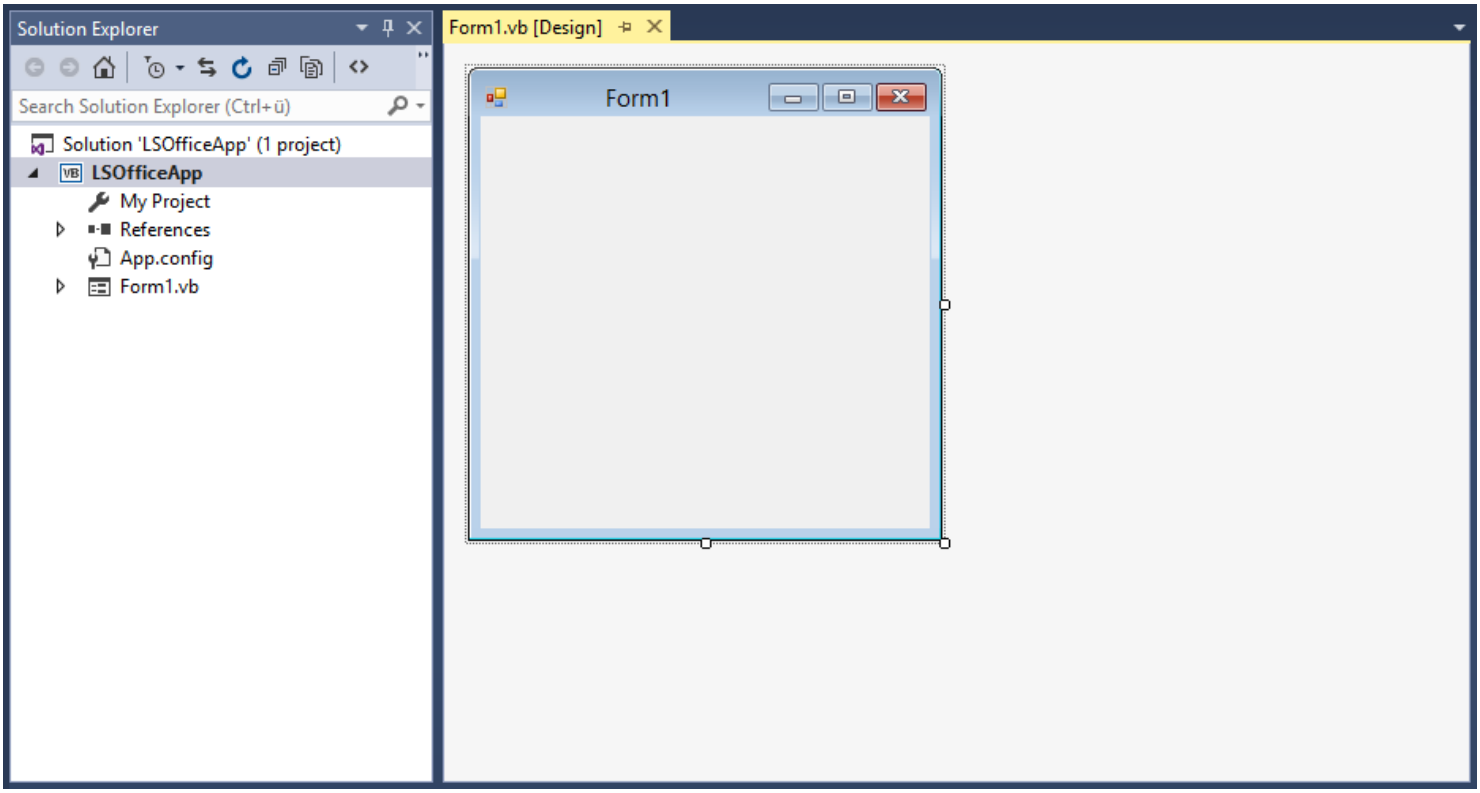
Step 1: Create a Windows Forms Application Project

The first step to create a new application is to create a Windows Forms application project.

To create a Windows Forms application project

1. Open **Visual Studio**.
2. On the menu bar, choose **File**, **New**, **Project**.
   The **New Project** dialog box opens.
3. Set target framework to **.Net Framework 4.6**.
4. Choose either **Visual C#** or **Visual Basic** in the **Installed Templates** list.
5. In the templates list, choose the **Windows Forms Application** icon.
6. In the **Name** text box, specify the name of the project. For this sample application, enter `LSOfficeApp`.
7. Click the **OK** button.

Visual Studio creates a solution for your program. A solution acts as a container for all of the projects and files needed by your program.



Step 2: Change Platform Target

The automation interface of **Labelstar Office** is a true 32-bit interface. For 64-bit systems, the application must be generated as a 32-bit application.

To change platform target (C#)

1. In **Solution Explorer**, right-click on **LSOfficeApp** project node and choose **Properties**.
2. Click the **Build** tab.
3. Choose **x86** from the **Platform target** list.

To change platform target (Visual Basic)

1. In **Solution Explorer**, right-click on **LSOfficeApp** project node and choose **Properties**.
2. Click the **Compile** tab.
3. Choose **x86** from the **Target CPU** list.

Note

When building your application with the wrong platform target, you may see the following error message:

There was a mismatch between the processor architecture of the project being built "MSIL" and the processor architecture of the reference "LSOffice", "x86". This mismatch may cause runtime failures. Please consider changing the targeted processor architecture of your project through the Configuration Manager so as to align the processor architectures between your project and references, or take a dependency on references with a processor architecture that matches the targeted processor architecture of your project.

Step 3: Include API in Project

The next step is to add a reference to the LSOffice Type Library to your development project. This allows your development environment to be aware of the LSOffice objects and their methods and properties.

To add a reference to LSOffice.dll to the project

1. In **Solution Explorer**, right-click on the **References** node and choose **Add Reference**.
2. In the **Reference Manager**, click **Browse**.
3. Go to the installation folder of **Labelstar Office** and select **LSOffice.dll**.
4. Click the **OK** button.

Step 4: Initialize Application Object

Now you are ready to create and initialize the Application object.

Add the following code to your *Form1* code file (Form1.cs or Form1.vb):

C#

```
LSOffice.Application objApp;

protected override void OnLoad (EventArgs e)
{
   // Initialize application
   objApp = new LSOffice.Application ();

   objApp.Initialize ();
   DisplayLastError ();

   if (objApp.License.IsTrialVersion)
      Text += " - Trial Version";

   base.OnLoad (e);
}

private bool DisplayLastError ()
{
   if (objApp.LastError.ErrorType == LSOffice.ErrorType.Success)
      return false;

   string title = "Error";

   if (objApp.LastError.ErrorType == LSOffice.ErrorType.Warning)
      title = "Warning";

   MessageBox.Show (objApp.LastError.Message, title);
   return true;
}
```

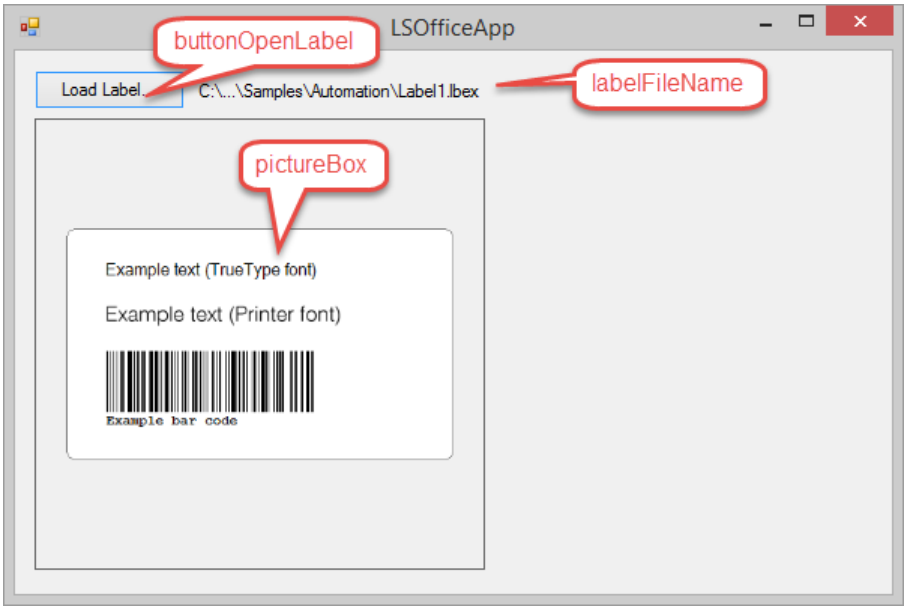

VB.Net

```
Dim objApp As LSOffice.Application

Protected Overrides Sub OnLoad(ByVal e As EventArgs)

   'Initialize application
   objApp = New LSOffice.Application

   objApp.Initialize()
   DisplayLastError()

   If (objApp.License.IsTrialVersion) Then
      Text += " - Trial Version"
   End If

   MyBase.OnLoad(e)

End Sub

Private Function DisplayLastError() As Boolean

   If (objApp.LastError.ErrorType = LSOffice.ErrorType.Success) Then
      Return False
   End If

   Dim title As String = "Error"

   If (objApp.LastError.ErrorType = LSOffice.ErrorType.Warning) Then
      title = "Warning"
   End If

   MsgBox(objApp.LastError.Message, vbOKOnly, title)
   Return True

End Function
```


Step 5: Write Code to Open Labels

In this step, you add code to open a label and show a label preview.

Add a Button, a Label, and a PictureBox control to the form.



Double-click the button to create a *buttonOpenLabel_Click* event handler, and add the following code:

C#

```
LSOffice.Label objLabel;

private void buttonOpenLabel_Click (object sender, EventArgs e)
{
   objLabel = null;

   OpenFileDialog dialog = new OpenFileDialog ();
   dialog.Filter = "Labels|*.lbex";

   if (dialog.ShowDialog () == DialogResult.OK)
      objLabel = objApp.OpenLabel (dialog.FileName);

   if (objLabel == null)
   {
      labelFileName.Text = "No label opened.";
      pictureBox.Image = null;
   }

   else
   {
      labelFileName.Text = objLabel.LabelPath;
      pictureBox.Image = objLabel.GetPreview ();
   }

   DisplayLastError ();
}
```

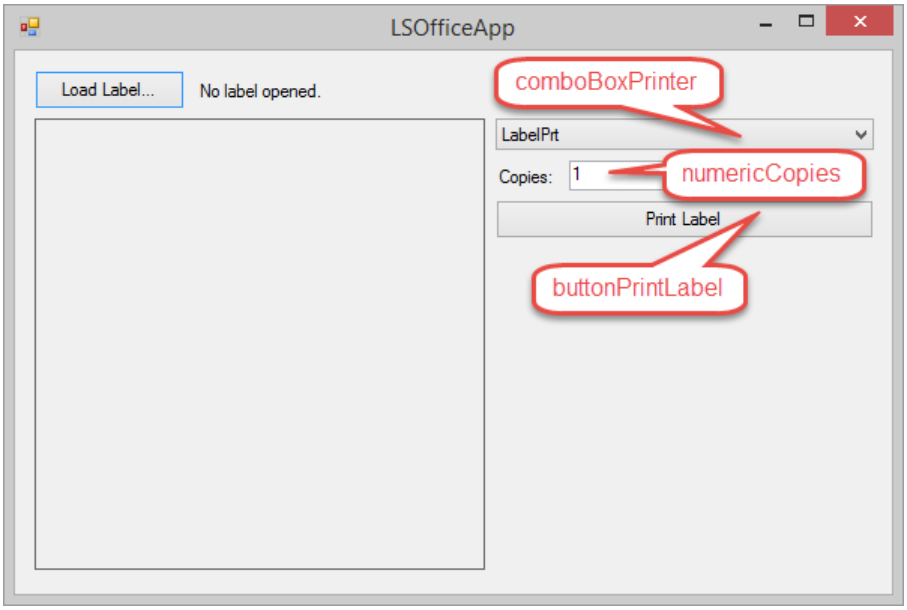

VB.Net

```
Dim objLabel As LSOffice.Label

Private Sub ButtonOpenLabel_Click(sender As Object, e As EventArgs) Handles ButtonOpenLabel.Click

   objLabel = Nothing

   Dim dialog As OpenFileDialog = New OpenFileDialog
   dialog.Filter = "Labels|*.lbex"

   If (dialog.ShowDialog() = DialogResult.OK) Then
      objLabel = objApp.OpenLabel(dialog.FileName)
   End If

   If (objLabel Is Nothing) Then
      LabelFileName.Text = "No label opened."
      PictureBox.Image = Nothing
   Else
      LabelFileName.Text = objLabel.LabelPath
      PictureBox.Image = objLabel.GetPreview
   End If

   DisplayLastError()

End Sub
```


Step 6: Write Code to Print Labels

In this step, you add code to select a printer, enter number of copies and print the opened label.

Add a ComboBox, a NumericUpDown, and a Button control to the form.



To see all printer drivers currently installed in Windows, add the following code:

C#

```
protected override void OnLoad (EventArgs e)
{
   // Initialize application

   ...

   // Insert installed printers
   foreach (string installedPrinter in System.Drawing.Printing.PrinterSettings.InstalledPrinters)
      comboBoxPrinter.Items.Insert (0, installedPrinter);

   // Set the combo box to the active printer
   comboBoxPrinter.SelectedItem = objApp.ActivePrinter;

   base.OnLoad (e);
}
```

VB.Net

```
Protected Overrides Sub OnLoad(ByVal e As EventArgs)

   'Initialize application

   ...

   'Insert installed printers
   For Each installedPrinter In Printing.PrinterSettings.InstalledPrinters
      ComboBoxPrinter.Items.Insert(0, installedPrinter)
   Next installedPrinter

   'Set the combo box to the active printer
   ComboBoxPrinter.SelectedItem = objApp.ActivePrinter

   MyBase.OnLoad(e)

End Sub
```


Double-click the combo box to create a *comboBoxPrinter_SelectedIndexChanged* event handler, and add the following code:

C#

```
private void comboBoxPrinter_SelectedIndexChanged (object sender, EventArgs e)
{
   if (objLabel == null)
      return;

   objLabel.ActivePrinter = (string)comboBoxPrinter.SelectedItem;
   pictureBox.Image = objLabel.GetPreview ();

   DisplayLastError ();
}
```

VB.Net

```
Private Sub ComboBoxPrinter_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
ComboBoxPrinter.SelectedIndexChanged

   If (objLabel Is Nothing) Then
      Return
   End If

   objLabel.ActivePrinter = ComboBoxPrinter.SelectedItem
   PictureBox.Image = objLabel.GetPreview

   DisplayLastError()

End Sub
```


Double-click the button to create a *buttonPrintLabel_Click* event handler, and add the following code:

C#

```
private void buttonPrintLabel_Click (object sender, EventArgs e)
{
   if (objLabel == null)
   {
      MessageBox.Show ("No label opened.");
      return;
   }

   objLabel.Copies = (int)numericCopies.Value;
   objLabel.Print ();
}
```

VB.Net

```
Private Sub ButtonPrintLabel_Click(sender As Object, e As EventArgs) Handles ButtonPrintLabel.Click

   If (objLabel Is Nothing) Then
      MsgBox("No label opened.")
      Return
   End If

   objLabel.Copies = NumericCopies.Value
   objLabel.Print()

End Sub
```


Step 7: Run Your Program and Try Other Features

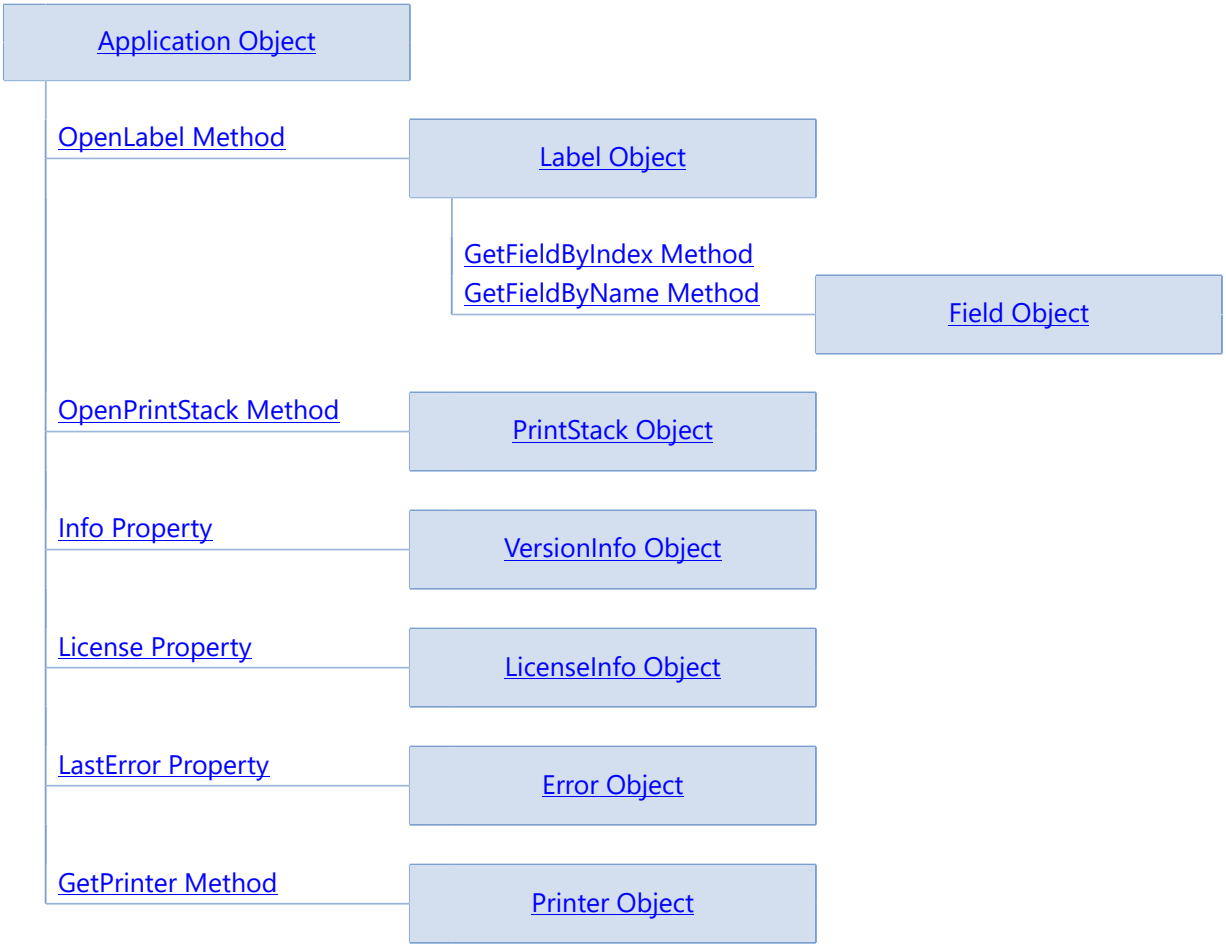
Your program is finished and ready to run. To learn more try to improve the program by modifying field data, and handling database labels.

The completed version of this sample and other sample projects can be found in the *<installation folder>\Automation* folder. If you intend to make changes to the sample projects you should copy this folder (and all its contents) to a user-accessible part of your file system.

Reference

This section will give you a brief overview over the functionalities of the API interface which are included in LSOffice.dll. These properties and methods can be used in customer applications, for controlling label printing.

The following diagram shows how the objects fit together:



Application Object

An **Application** object represents the entire **Labelstar Office** application and enables you to open, save, and print labels, and to modifiy the label content.

See also, Using the Application Object.

Properties

	Name	Description
	ActivePrinter	Returns the name of the active printer.
	HasError	Gets a value that indicates whether an error occurred during the last call to a method or property.
	Info	Refers to the VersionInfo object representing the version information.
	IsInitialized	Gets a value that indicates whether the Initialize method has been called.
	LabelDir	Returns the path of the current label folder.
	LastError	Retrieves the calling method's or property's last-error code value.
	License	Refers to the LicenseInfo object representing the license information.

Methods

	Name	Description
	GetPrinter (string)	Returns a Printer object.
	Import (string, string)	Imports a Labelstar Plus label.
	Initialize ()	Initialises the **Application** object on the basis of the current program settings.
	OpenLabel (string)	Opens the specified label and gets a Label object.
	OpenPrintStack (string)	Opens the specified print stack and gets a PrintStack object.

ActivePrinter Property

Returns the name of the active printer. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.ActivePrinter
```

Type

```
String
```

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
   MessageBox.Show (objApp.LastError.Message, "Error");
else
   MessageBox.Show ($"The name of the active printer is {objApp.ActivePrinter}", "Demo");
```

Example (VB.Net)

```
Dim objApp As LSOffice.Application

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize ()

If (objApp.HasError) Then
   MsgBox (objApp.LastError.Message, vbOKOnly, "Error")
Else
   MsgBox ("The name of the active printer is " & objApp.ActivePrinter, vbOKOnly, "Demo")
End If
```

See also

- Application Object

GetPrinter Method

Returns a Printer object.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Application.GetPrinter (printerName)
```

Parameters

printerName
Type: `String`
The name of the printer.

Return Value

Type: `LSOffice.Printer`
Returns a Printer object or **null**, if the printer name is invalid. To get extended error information, check the value of the LastError property.

Remarks

When using a Printer object, make sure you call Printer.Dispose when you are done.

See also

- Using the Printer Object
- Application Object

HasError Property

Gets a value that indicates whether an error occurred during the last call to a method or property. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.HasError
```

Type

```
Boolean
```

Remarks

For more information, see LastError property.

See also

- Using the Application Object
- Application Object

Import Method

Imports a Labelstar Plus label.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.50.1010

Syntax

```
Application.Import (sourceFileName, destFileName)
```

Parameters

sourceFileName
Type: `String`
The name of the file to import.

destFileName
Type: `String`
The new path and name of the file.

Remarks

Check LastError property to see if the function was completed successfully,

Info Property

Refers to the VersionInfo object representing the version information. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.Info
```

Type

```
LSOffice.VersionInfo
```

See also

- Application Object

Initialize Method

Initialises the Application object on the basis of the current program settings.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.Initialize ()
```

Remarks

This method ensures that the current instance is properly initialized before it is used to open labels. It should only be called once. This method sets the IsInitialized property to **true**.

Check LastError property to see if the function was completed successfully,

---

See also

- Using the Application Object
- Application Object

IsInitialized Property

Gets a value that indicates whether the Initialize method has been called. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.IsInitialized
```

Type

```
Boolean
```

Remarks

Use this property to check whether the Application object has already been initialized. The Initialize method should be called once and only once.

See also

- Application Object

LabelDir Property

Returns the path of the current label folder. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.LabelDir
```

Type

```
String
```

See also

- Application Object

LastError Property

Retrieves the calling method's or property's last-error code value. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.LastError
```

Type

LSOffice.Error

Remarks

After method or property calls on all Automation objects, this property is updated with the return status information of the call. To check whether the request has been successful, check HasError property.

For more information, see Error Codes and Messages.

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
}

else
{
   // Open label
   LSOffice.Label label = objApp.OpenLabel ("C:\\Label1.lbex");

   if (label == null)
      MessageBox.Show (objApp.LastError.Message, "Error");
}
```

Example (VB.Net)

```
Dim objApp As LSOffice.Application
Dim objLabel As LSOffice.Label

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize()

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
Else
   'Open label
   objLabel = objApp.OpenLabel("C:\\Label1.lbex")

   If (objLabel Is Nothing) Then
      MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   End If
End If
```

See also

- Using the Application Object
- Application Object

License Property

Refers to the LicenseInfo object representing the license information. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.License
```

Type

`LSOffice.LicenseInfo`

Remarks

In the trial version an evaluation mode watermark will be put onto each image and all 'e' are replaced by 'x' and all '5' by '0'.

See also

- Application Object

OpenLabel Method

Opens the specified label.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Application.OpenLabel (path)
```

Parameters

path
Type: `String`
Label path

Return Value

Type: `LSOffice.Label`
Returns a Label object if opened successfully, otherwise **null**. To get extended error information, check the value of the LastError property.

See also

- Using the Label Object
- Application Object

OpenPrintStack Method

Opens the specified print job and gets a PrintStack object.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.50.1010

Syntax

```
Application.OpenPrintStack (path)
```

Parameters

path
Type: `String`
File path

Return Value

Type: `LSOffice.PrintStack`
Returns a PrintStack object if opened successfully, otherwise **null**. To get extended error information, check the value of the LastError property.

Error Object

An **Error** object holds information about the current error state. You can get a reference to an **Error** object through the Application.LastError property.

Properties

	Name	Description
	Details	Detailed message describing the error occurred.
	ErrorCode	Numeric code indicating which error occurred.
	ErrorType	Code indicating the type of the error occurred.
	Message	Message describing the error occurred.

Details Property

Detailed message describing the error occurred. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Error.Details
```

Type

```
String
```

Remarks

Returns a representation of the current error that is intended to be understood by humans. The detailed message obtains the Message property, further information about the error, and the stack trace.

See also

- Error Object

ErrorCode Property

Numeric code indicating which error occurred. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Error.ErrorCode
```

Type

```
Integer
```

Remarks

If this property is 0, it indicates that no error occured during the last call of a method or property. For more information, see Error Codes and Messages.

See also

- Error Object

ErrorType Property

Code indicating the type of the error occurred. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Error.ErrorType
```

Type

```
LSOffice.ErrorType
```

	Name	Value	Description
	Success	0 (0x00)	No error occurred.
	Warning	1 (0x01)	The call succeeded, but there is a potentional problem.
	Error	2 (0x02)	The call failed.

See also

- Using the Application Object
- Error Object

Message Property

Message describing the error occurred. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Error.Message
```

Type

```
String
```

Remarks

Returns a representation of the current error that is intended to be understood by humans. If there is no error, an empty string ("") is returned. For more information, see Error Codes and Messages.

See also

- Using the Application Object
- Error Object

Field Object

A **Field** object represents a field on the label. It can be used to get information about the field, and to get and set its content and properties. You can get a reference to a **Field** object through the Label.GetFieldByIndex or Label.GetFieldByName method.

See also, Using the Label Object.

Properties

	Name	Description
🔒	FieldName	Gets the name of the field.
🔒	Locked	Gets a value that indicates whether the field can be modified.
	Printable	Gets or sets a value that indicates whether the field is printed.

Methods

	Name	Description
	GetContent ()	Gets the content of the field.
	GetPropertyValue (string)	Gets the value of the specified property.
	SetContent (string)	Sets the content of the field.
	SetPropertyValue (string, object)	Sets the value of the specified property.

FieldName Property

Gets the name of the field. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.FieldName
```

Type

```
String
```

See also

- Field Object

GetContent Method

Gets the content of the field.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.GetContent ()
```

Return Value

Type: `String`
If the method succeeds it returns the content of the field, otherwise an empty string (""). Check Application.LastError property for information about possible errors.

See also

- Field Object

GetPropertyValue Method

Gets the value of the specified property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.GetPropertyValue (propertyName)
```

Parameters

propertyName
Type: `String`
The name of the property. The list of possible property names can be found here.

Return Value

Type: `Object`
If the method succeeds it returns the value of the property, otherwise **null**. Check Application.LastError property for information about possible errors.

See also

- Field Object

Property Names

The following table describes some possible property names that can be set on fields.

Property name	Description
HumanReadable	**Bar Code only** Type: `Boolean` false or 0: Don't show human readable text true or 1: Show human readable text
Locked	Type: `Boolean` false or 0: Field is not locked true or 1: Field is locked See also, Locked Property
Printable	Type: `Boolean` false or 0: Field is not printed true or 1: Field is printed See also, Printable Property
TextAlignment	**Text/Bar Code only** Type: `Integer` 0: Left 1: Center 2: Right 3: Justify

Locked Property

Gets a value that indicates whether the field can be modified. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.Locked
```

Type

```
Boolean
```

See also

- Field Object

Printable Property

Gets or sets a value that indicates whether the field is printed.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.Printable
```

Type

```
Boolean
```

See also

- Field Object

SetContent Method

Sets the content of the field.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.SetContent (content)
```

Parameters

content
Type: `String`
The new content of the field.

Remarks

Check Application.LastError property for information about possible errors.

See also

- Using the Label Object
- Field Object

SetPropertyValue Method

Sets the value of the specified property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Field.SetPropertyValue (propertyName, value)
```

Parameters

propertyName
Type: `String`
The name of the property. The list of possible property names can be found here.

value
Type: `Object`
The value to set the property to.

Remarks

Check Application.LastError property for information about possible errors.

See also

- Field Object

Label Object

A **Label** object represents an opened label. You can get a reference to a **Label** object by calling the Application.OpenLabel method.

See also, Using the Label Object.

Properties

	Name	Description
	ActivePrinter	Gets or sets the name of the active printer.
	Copies	Gets or sets the number of copies to print.
	CurrentRecord	Gets or sets the one-based index of the current record to be printed.
🔓	FieldCount	Gets the number of fields defined on the label.
🔓	FieldNames	Gets the list of fields defined on the label.
🔓	IsDataAvailable	Determines if there are database fields defined on the label.
🔓	LabelPath	Gets the path to the opened label.
🔓	MaxRecord	Returns the maximum number of records in the database.
🔓	Modified	Gets a value that indicates that the label has been modified.
	PageName	Gets or sets the current page name.
	PrintOptions	Gets or sets the print options.

Methods

	Name	Description
	GetFieldByIndex (int)	Gets the field with the given index.
	GetFieldByName (string)	Searches for the field with the specified name.
	GetPreview ()	Retrieves a preview image of the current label content.
	GetPropertyValue (string)	Gets the value of the specified property.
	Print ()	Prints the label using the current print settings.
	Print (int)	Prints the label with specified number of copies.
	Print (int, PrintOptions)	Prints the label with specified number of copies and print options.
	PrintToFile (string)	Prints the label to a file using the current print settings.
	PrintToFile (string, int)	Prints the label to a file with specified number of copies.
	PrintToFile (string, int, PrintOptions)	Prints the label to a file with specified number of copies and print options.
	Save ()	Saves changes to the label.
	SaveAs (string)	Saves changes to the label in a different file.
	SavePreview (string)	Saves a preview picture of the current label in a file as bitmap (BMP).
	SavePreview (string, ImageFormat)	Saves a preview picture of the current label in a file.
	SelectRecord (string)	Sets the current record to the first record matching the filter expression.
	SelectRecord (string, string)	Sets the current record to the first record matching the filter expression from the specified database.
	SetPropertyValue (string, object)	Sets the value of the specified property.

ActivePrinter Property

Gets or sets the name of the active printer.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.ActivePrinter
```

Type

```
String
```

See also

- Label Object

Copies Property

Gets or sets the number of copies to print.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Label.Copies
```

Type

```
Integer
```

Remarks

If number of copies > 0 the **Print** dialog box will not be shown.

See also

- Using the Label Object
- Label Object

CurrentRecord Property

Gets or sets the one-based index of the current record to be printed.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.CurrentRecord
```

Type

```
Integer
```

Remarks

To check if database fields are defined on the label use IsDataAvailable property.

See also

- Label Object

FieldCount Property

Gets the number of fields defined on the label. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.FieldCount
```

Type

```
Integer
```

See also

- Label Object

FieldNames Property

Gets the list of fields defined on the label. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.FieldNames
```

Type

```
String[]
```

See also

- Label Object

GetFieldByIndex Method

Gets the field with the given index.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.GetField (index)
```

Parameters

index
Type: `Integer`
Zero-based index of the field.

Return Value

Type: `LSOffice.Field`
This method returns a Field object on the indicated field, if found; otherwise, **null**. To get extended error information, check the value of the Application.LastError property.

See also

- Label Object

GetFieldByName Method

Searches for the field with the specified name.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.GetField (fieldName)
```

Parameters

fieldName
Type: `String`
The string containing the name of the field to get.

Return Value

Type: `LSOffice.Field`
Returns a Field object representing the field with the specified name, if found; otherwise, **null**. To get extended error information, check the value of the Application.LastError property.

See also

- Using the Label Object
- Label Object

GetPreview Method

Retrieves a preview image of the current label content.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.GetPreview ()
```

Return Value

Type: `Bitmap`
Returns a **Bitmap** object representing the preview, if a valid label is opened; otherwise, **null**.

See also

- Label Object

GetPropertyValue Method

Gets the value of the specified property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Syntax

```
Label.GetPropertyValue (propertyName)
```

Parameters

propertyName
Type: `String`
The name of the property. The list of possible property names can be found here.

Return Value

Type: `Object`
If the method succeeds it returns the value of the property, otherwise **null**. Check Application.LastError property for information about possible errors.

See also

- Label Object

Property Names

The following table describes some possible property names that can be set on labels.

Property name	Description
LabelRotation	Type: `Integer` 0, 90, 180, 270
LabelType	Type: `Integer` 0: Adhesive labels 1: Continuous labels
PrinterName	Type: `String`

IsDataAvailable Property

Determines if there are database fields defined on the label. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.IsDataAvailable
```

Type

```
Boolean
```

See also

- Label Object

LabelPath Property

Gets the path to the opened label. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.LabelPath
```

Type

```
String
```

See also

- Label Object

MaxRecord Property

Returns the maximum number of records in the database. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.MaxRecord
```

Type

```
Integer
```

Remarks

Returns the maximum number of records in the database, or 0 if no database fields are defined on the label. To check if database fields are defined on the label use IsDataAvailable property.

See also

- Label Object

Modified Property

Gets a value that indicates that the label has been modified. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.Modified
```

Type

```
Boolean
```

See also

- Label Object

PageName Property

Gets or sets the current page name.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Syntax

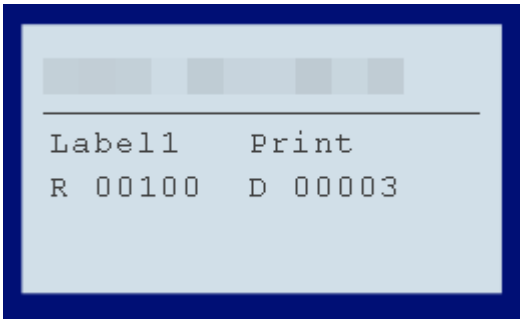
```
Label.PageName
```

Type

```
String
```

Remarks

The page name is displayed in the printer display during printing.



See also

- Label Object

Print Method

Prints the label using the current print settings.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Label.Print ()
```

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Print Method (int)
- Print Method (int, PrintOptions)
- Using the Label Object
- Label Object

Print Method (int)

Prints the label with specified number of copies.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.40.1010

Syntax

```
Label.Print (copies)
```

Parameters

copies
Type: `Integer`
Number of copies to print. If *copies* > 0 the **Print** dialog box will not be shown.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Print Method
- Print Method (int, PrintOptions)
- Label Object

Print Method (int, PrintOptions)

Prints the label with specified number of copies and print options.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.Print (copies, options)
```

Parameters

copies
Type: `Integer`
Number of copies to print. If *copies* > 0 the **Print** dialog box will not be shown.

options
Type: `LSOffice.PrintOptions`
Print options. For a description of the options, see PrintOptions property.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Print Method
- Print Method (int)
- Label Object

PrintOptions Property

Gets or sets the print options.

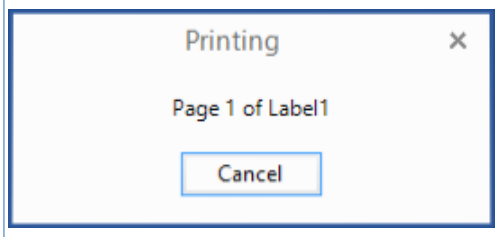
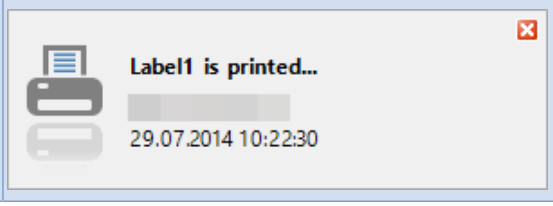
Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Label.PrintOptions
```

Type

```
LSOffice.PrintOptions
```

	Name	Value	Description
	Default	0 (0x00)	Default settings
	PrintCurrentRecord	1 (0x01)	Prints the current selected record. **Select Record** dialog box will no be shown.
	PrintAllRecords	2 (0x02)	Prints all records. **Select Record** dialog box will not be shown.
	ShowPrintingDialog	4 (0x04)	Shows a **Printing** dialog box as long as labels are sent to the printer. If you decide to cancel the printing process - and you're quick enough - you can click the **Cancel** button in this **Printing** dialog box to stop the operation. 
	ShowNotificationMessage	8 (0x08)	Shows a notification at the bottom-right of your screen when a label is printed. 

Remarks

This enumeration allows a bitwise combination of its member values.

The Print and PrintToFile methods use this enumeration too.

For more information and a detailed example, see Label.SelectRecord method.

See also

- Label Object

PrintToFile Method (string)

Prints the label to a file using the current print settings.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Label.PrintToFile (fileName)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- PrintToFile Method (string, int)
- PrintToFile Method (string, int, PrintOptions)
- Label Object

PrintToFile Method (string, int)

Prints the label to a file with specified number of copies.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.40.1010

Syntax

```
Label.PrintToFile (fileName, copies)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

copies
Type: `Integer`
Number of copies to print. If *copies* > 0 the **Print** dialog box will not be shown.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- PrintToFile Method (string)
- PrintToFile Method (string, int, PrintOptions)
- Label Object

PrintToFile Method (string, int, PrintOptions)

Prints the label to a file with specified number of copies and print options.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Syntax

```
Label.PrintToFile (fileName, copies, options)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

copies
Type: `Integer`
Number of copies to print. If *copies* > 0 the **Print** dialog box will not be shown.

options
Type: `LSOffice.PrintOptions`
Print options. For a description of the options, see PrintOptions property.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- PrintToFile Method (string)
- PrintToFile Method (string, int)
- Label Object

Save Method

Saves changes to the label.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.Save ()
```

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- SaveAs Method
- Label Object

SaveAs Method

Saves changes to the label in a different file.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.SaveAs (fileName)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Save Method
- Label Object

SavePreview Method (string)

Saves a preview picture of the current label in a file as bitmap (BMP).

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.40.1010

Syntax

```
Label.SavePreview (fileName)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Label Object

SavePreview Method (string, ImageFormat)

Saves a preview picture of the current label in a file.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Syntax

```
Label.SavePreview (fileName, format)
```

Parameters

fileName
Type: `String`
The name of the file to be saved.

format
Type: `LSOffice.ImageFormat`
The image format to be saved.

	Name	Value	Description
	Bmp	0 (0x00)	Saves the image as a bitmap (BMP).
	Gif	1 (0x01)	Saves the image using the GIF image format.
	Jpeg	2 (0x02)	Saves the image using the Joint Photographic Experts Group (JPEG) image format.
	Png	3 (0x03)	Saves the image using the W3C Portable Network Graphics (PNG) image format.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Label Object

SelectRecord Method (string)

Sets the current record to the first record matching the filter expression.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
Label.SelectRecord (filterExpression)
```

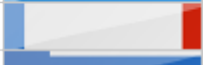
Parameters

filterExpression
Type: `String`
The criteria to use to filter the records. For examples on how to filter records, see Filter Expression Syntax.

Remarks

Check Application.LastError property to see if the method was completed successfully.

Example Database (Europe.accdb)

ID	Name	Capital	Area	Population	NativeName	Flag
1	Albania	Tirana	28748	3560000	Shqipëria	
2	Andorra	Andorra la Vella	464	72300	Andorra	
3	Belgium	Brussels	30518	1030000	Belgique	
4	Bosnia and Her...	Sarajevo	51129	3800000	Босна и Херце...	
5	Bulgaria	Sofia	110994	7810000	България	
6	Denmark	Copenhagen	43093	5410000	Danmark	
7	Germany	Berlin	357114	82220000	Deutschland	
8	Estonia	Tallinn	45227	1350000	Eesti	
9	Finland	Helsinki	338145	5220000	Suomi	
10	France	Paris	547026	60660000	France	
11	Greece	Athens	131957	10670000	Ελλάς	
12	United Kingdom	London	244820	60440000	United Kingdom	
13	Ireland	Dublin	70284	4040000	Éire	

Example (C#)

```
LSOffice.Application objApp;

// Initialize application
objApp = new LSOffice.Application ();
objApp.Initialize ();

if (objApp.HasError)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

// Open label
LSOffice.Label objLabel = objApp.OpenLabel ("C:\\Label1.lbex");

if (objLabel == null)
{
   MessageBox.Show (objApp.LastError.Message, "Error");
   return;
}

objLabel.PrintOptions = LSOffice.PrintOptions.PrintCurrentRecord;
objLabel.Copies = 1;

// Print original label
objLabel.Print ();

// Select record
objLabel.CurrentRecord = 10;

if (objApp.HasError)
   MessageBox.Show (objApp.LastError.Message, "Error");
else
   objLabel.Print ();

// Select record
string selectionString = "Deutschland";
objLabel.SelectRecord ($"NativeName LIKE '{selectionString}'");

if (objApp.HasError)
   MessageBox.Show (objApp.LastError.Message, "Error");
else
   objLabel.Print ();
```


Example (VB.NET)

```
Dim objApp As LSOffice.Application
Dim objLabel As LSOffice.Label

'Initialize application
objApp = New LSOffice.Application
objApp.Initialize()

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

'Open label
objLabel = objApp.OpenLabel("C:\\Label1.lbex")

If (objLabel Is Nothing) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
   Return
End If

objLabel.PrintOptions = LSOffice.PrintOptions.PrintCurrentRecord
objLabel.Copies = 1

'Print original label
objLabel.Print()

'Select record
objLabel.CurrentRecord = 10

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
Else
   objLabel.Print()
End If

'Select record
Dim selectionString = "Deutschland"
objLabel.SelectRecord("NativeName LIKE'" & selectionString & "'")

If (objApp.HasError) Then
   MsgBox(objApp.LastError.Message, vbOKOnly, "Error")
Else
   objLabel.Print()
End If
```

See also

- Label Object

Filter Expression Syntax

This example describes syntax of filter expression. It shows how to correctly build aexpression string (without „SQL injection“) using methods to escape values.

Column Names

If a column name contains any of these special characters ~ ( ) # \ / = > < + - * % & | ^ ' " [ ], you must enclose the column name within square brackets [ ]. If a column name contains right bracket] or backslash \, escape it with backslash (\] or \\).

Example	Description
"id = 10"	No special character in column name "id".
"$id = 10"	No special character in column name "$id".
"[#id] = 10"	Special character "#" in column name "#id".
"[[id\]] = 10"	Special characters in column name "[id]".

Literals

String values are enclosed within single quotes ' '. If the string contains single quote ', the quote must be doubled.

Example	Description
"Name = 'John'"	String value.
"Name = 'John ''A'''"	String with single quotes "John 'A'".

Number values are not enclosed within any characters. The values should be formatted in English locale format.

Example	Description
"Year = 2008"	Integer value.
"Price = 1199.9"	Float value.

Date values are enclosed within sharp characters # #. The date format is the same as for English culture.

Example	Description
"Date = #12/31/2008#"	Date value (time is 00:00:00).
"Date = #2008-12-31#"	Also this format is supported.
"Date = #12/31/2008 16:44:58#"	Date and time value.

Alternatively you can enclose all values within single quotes ' '. It means you can **use string values** for numbers or date time values. In this case the current culture is used to convert the string to the specific value.

Example	Description
"Date = '12/31/2008 16:44:58'"	Current culture is English.
"Date = '31.12.2008 16:44:58'"	Current culture is German.
"Price = '1199.90'"	Current culture is English.
"Price = '1199,90'"	Current culture is German.

Comparison Operators

Equal, not equal, less, greater operators are used to include only values that suit to a comparison expression. You can use these operators = <> < <= > >=.

Note: String comparison is culture-sensitive.

Example	Description
"Num = 10"	Number is equal to 10.
"Date < #1/1/2008#"	Date is less than 1/1/2008.
"Name <> 'John'"	String is not equal to 'John'.
"Name >= 'Jo'"	String comparison.

Operator IN is used to include only values from the list. You can use the operator for all data types, such as numbers or strings.

Example	Description
"Id IN (1, 2, 3)"	Integer values.
"Price IN (1.0, 9.9, 11.5)"	Float values.
"Name IN ('John', 'Jim', 'Tom')"	String values.
"Date IN (#12/31/2008#, #1/1/2009#)"	Date time values.
"Id NOT IN (1, 2, 3)"	Values not from the list.

Operator LIKE is used to include only values that match a pattern with wildcards. **Wildcard** character is * or %, it can be at the beginning of a pattern '*value', at the end 'value*', or at both '*value*'. Wildcard in the middle of a patern 'va*lue' is **not allowed**.

Example	Description
"Name LIKE 'j*'"	Values that start with 'j'.
"Name LIKE '%jo%'"	Values that contain 'jo'.
"Name NOT LIKE 'j*'"	Values that don't start with 'j'.

If a pattern in a LIKE clause contains any of these special characters * % [ ], those characters must be escaped in brackets [ ] like this [*], [%], [[] or []].

Example	Description
"Name LIKE '[*]*'"	Values that starts with '*'.
"Name LIKE '[[]*'"	Values that starts with '['.

Boolean Operators

Boolean operators **AND, OR** and **NOT** are used to concatenate expressions. Operator NOT has precedence over AND operator and it has precedence over OR operator.

Example	Description
"City = 'Tokyo' AND (Age < 20 OR Age > 60)"	Operator AND has precedence over OR operator, parenthesis are needed.
"City <> 'Tokyo' AND City <> 'Paris'"; "NOT City = 'Tokyo' AND NOT City = 'Paris'"; "NOT (City = 'Tokyo' OR City = 'Paris')"; "City NOT IN ('Tokyo', 'Paris')";	These examples do the same.

Arithmetic and String Operators

Arithmetic operators are addition +, subtraction -, multiplication *, division / and modulus %.

Example	Description
"MotherAge - Age < 20"	People with young mother.
"Age % 10 = 0"	People with decennial birthday.

There is also one **string** operator **concatenation +**.

Parent-Child Relation Referencing

A **parent table** can be referenced in an expression using parent column name with `Parent.` prefix. A column in a **child table** can be referenced using child column name with `Child.` prefix.

The reference to the child column must be in an aggregate function because child relationships may return multiple rows. For example expression `SUM(Child.Price)` returns sum of all prices in child table related to the row in parent table.

If a table has more than one child relation, the prefix must contain relation name. For example expression `Child(OrdersToItemsRelation).Price` references to column Price in child table using relation named OrdersToItemsRelation.

Aggregate Functions

There are supported following aggregate functions **SUM, COUNT, MIN, MAX, AVG** (average), **STDEV** (statistical standard deviation) and **VAR** (statistical variance).

Example	Description
"Salary > AVG(Salary)"	Select people with above-average salary.
"COUNT(Child.IdOrder) > 5"	Select orders which have more than 5 items.
"SUM(Child.Price) >= 500"	Select orders which total price (sum of items prices) is greater or equal $500.

Functions

There are also supported following functions. Detailed description can be found here Filter Expression Functions.

- `CONVERT` – converts particular expression to a specified type
- `LEN` – gets the length of a string
- `ISNULL` – checks an expression and either returns the checked expression or a replacement value
- `IIF` – gets one of two values depending on the result of a logical expression
- `TRIM` – removes all leading and trailing blank characters like \r, \n, \t, , ‘
- `SUBSTRING` – gets a sub-string of a specified length, starting at a specified point in the string

---

See also

- Filter Expression Functions
- SelectRecord Method
- Label Object

Filter Expression Functions

The following functions are supported:

CONVERT

Description	Converts particular expression to a specified type.
Syntax	CONVERT (*expression*, *type*)
Arguments	*expression* -- The expression to convert. *type* -- The type to which the value will be converted.
Example	Expression = "Convert (total, 'System.Int32')"

All conversions are valid with the following exceptions: **Boolean** can be coerced to and from **Byte**, **SByte**, **Int16**, **Int32**, **Int64**, **UInt16**, **UInt32**, **UInt64**, **String** and itself only. **Char** can be coerced to and from **Int32**, **UInt32**, **String**, and itself only. **DateTime** can be coerced to and from **String** and itself only. **TimeSpan** can be coerced to and from **String** and itself only.

LEN

Description	Gets the length of a string.
Syntax	LEN (*expression*)
Arguments	*expression* -- The expression to evaluated.
Example	Expression = "Len (item)"

ISNULL

Description	Checks an expression and either returns the checked expression or a replacement value.
Syntax	ISNULL (*expression*, *replacementvalue*)
Arguments	*expression* -- The expression to check. *replacementvalue* -- If expression is **Nothing**, replacementvalue is returned.
Example	Expression = "IsNull (price, -1)"

IIF

Description	Gets one of two values depending on the result of a logical expression.
Syntax	IIF (*expression*, *truepart*, *falsepart*)
Arguments	*expression* -- The expression to evaluated. *truepart* -- The value to return if the expression is true. *falsepart* -- The value to return if the expression is false.
Example	Expression = "IIF (total>1000, 'expensive', 'dear')"

TRIM

Description	Removes all leading and trailing blank characters like \r, \n, \t, ' '.
Syntax	TRIM (*expression*)
Arguments	*expression* -- The expression to trim.

SUBSTRING

Description	Gets a sub-string of a specified length, starting at a specified point in the string.
Syntax	SUBSTRING (*expression*, *start*, *length*)
Arguments	*expression* -- The source string for the substring. *start* -- Integer that specifies where the substring starts. *length* -- Integer that specifies the length of the substring.
Example	Expression = "SubString (phone, 7, 8)"

See also

- Filter Expression Syntax
- SelectRecord Method
- Label Object

SelectRecord Method (string, string)

Sets the current record to the first record matching the filter expression from the specified database.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.50.1010

Syntax

```
Label.SelectRecord (databaseName, filterExpression)
```

Parameters

databaseName
Type: `String`
Specifies the name of the database.

filterExpression
Type: `String`
The criteria to use to filter the records. For examples on how to filter records, see Filter Expression Syntax.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Label Object

SetPropertyValue Method

Sets the value of the specified property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.20.1040

Syntax

```
Label.SetPropertyValue (propertyName, value)
```

Parameters

propertyName
Type: `String`
The name of the property. The list of possible property names can be found here.

value
Type: `Object`
The value to set the property to.

Remarks

Check Application.LastError property for information about possible errors.

See also

- Label Object

LicenseInfo Object

A **LicenseInfo** object represents the license information. You can get a reference to a **LicenseInfo** object through the Application.License property.

Properties

	Name	Description
	IsTrialVersion	Gets a value that indicates whether the application is in trial mode.
	LicenseKey	Gets the license key used to activate **Labelstar Office**.

IsTrialVersion Property

Gets a value that indicates whether the application is in trial mode. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
LicenseInfo.IsTrialVersion
```

Type

```
Boolean
```

Remarks

In the trial version an evaluation mode watermark will be put onto each image and all 'e' are replaced by 'x' and all '5' by '0'.

See also

- LicenseInfo Object

LicenseKey Property

Gets the license key used to activate **Labelstar Office**. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
LicenseInfo.LicenseKey
```

Type

```
String
```

See also

- LicenseInfo Object

Printer Object

A **Printer** object represents a printer installed on your computer. You can get a reference to a **Printer** object by calling the Application.GetPrinter method.

See also, Using the Printer Object.

Properties

	Name	Description
	Encoding	Gets the character encoding of the printer.
	IsCVPrinter	Determines if the selected printer is a valid Carl Valentin label printer.
	PrinterName	Gets the name of the printer.

Methods

	Name	Description
	Dispose ()	Releases memory allocated by the **Printer** object.
	Execute (PrintCommand)	Executes predefined print commands.
	Execute (string)	Executes custom print commands.
	GetStatus ()	Retrieves the status of the printer.
	GetStatus (out string)	Retrieves the status of the printer, along with additional information.
	SendBytes (byte[])	Sends a number of bytes to the printer.

Dispose Method

Releases memory allocated by the **Printer** object.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.Dispose ()
```

Remarks

Call **Dispose** when you are finished using the **Printer** object. Otherwise, the memory allocated by the **Printer** object will not be freed.

See also

- Using the Printer Object
- Printer Object

Encoding Property

Gets the character encoding of the printer. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.Encoding
```

Type

```
System.Text.Encoding
```

See also

- Printer Object

IsCVPrinter Property

Determines if the selected printer is a valid Carl Valentin label printer. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.IsCVPrinter
```

Type

```
Boolean
```

Remarks

If the printer is a valid Carl Valentin label printer special print commands can be sent to the printer, e.g. to stop, continue or cancel a print job.

See also

- Execute Method (PrintCommand)
- Execute Method (string)
- Using the Printer Object
- Printer Object

Execute Method (PrintCommand)

Executes predefined print commands. Carl Valentin label printers only.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.Execute (command)
```

Parameters

command
Type: `LSOffice.PrintCommand`

	Name	Value	Description
	LabelFeed	1 (0x01)	Releases a label feed.
	TestPrint	2 (0x02)	Releases a test print.
	StatusPrint	3 (0x03)	Prints a status print report.
	MeasureLabel	4 (0x04)	Measures the labels.
	StopPrinting	5 (0x05)	Pause printing.
	ContinuePrinting	6 (0x06)	Resume printing.Dru
	CancelPrinting	7 (0x07)	Cancel printing.

Return Value

Type: `Boolean`
Returns **true** if the command was transmitted successfully to the printer; otherwise, **false**. To get extended error information, check the value of the Application.LastError property.

See also

- Execute Method (string)
- Using the Printer Object
- Printer Object

Execute Method (string)

Executes custom print commands. Carl Valentin label printers only.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.Execute (command)
```

Parameters

command
Type: `String`
Custom print command.

Example: `"(SOH)FF----r(ETB)"`
Each command begins with a start character (SOH), followed by the data characters, and finally a stop character (ETB).

Please contact your local dealer to get a documentation of the supported print commands.

Return Value

Type: `Boolean`
Returns **true** if the command was transmitted successfully to the printer; otherwise, **false**. To get extended error information, check the value of the Application.LastError property.

See also

- Execute Method (PrintCommand)
- Using the Printer Object
- Printer Object

GetStatus Method

Retrieves the status of the printer.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.GetStatus ()
```

Return Value

Type: `LSOffice.PrinterStatus`

	Name	Value	Description
	Other	1 (0x01)	
	Unknown	2 (0x02)	
	Idle	3 (0x03)	
	Printing	4 (0x04)	
	Warmup	5 (0x05)	
	Stopped	6 (0x06)	
	Offline	7 (0x07)	
	Error	8 (0x08)	

See also

- Using the Printer Object
- Printer Object

GetStatus Method (out string)

Retrieves the status of the printer, along with additional information.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.GetStatus (out message)
```

Parameters

message [out]
Type: `String`
Additional status information, e.g. error message displayed in the printer display.

Return Value

Type: `LSOffice.PrinterStatus`

	Name	Value	Description
	Other	1 (0x01)	
	Unknown	2 (0x02)	
	Idle	3 (0x03)	
	Printing	4 (0x04)	
	Warmup	5 (0x05)	
	Stopped	6 (0x06)	
	Offline	7 (0x07)	
	Error	8 (0x08)	

See also

- Using the Printer Object
- Printer Object

PrinterName Property

Gets the name of the printer. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.PrinterName
```

Type

```
String
```

See also

- Printer Object

SendBytes Method

Send a number of bytes to the printer.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.20.1010

Syntax

```
Printer.SendBytes (bytes)
```

Parameters

bytes
Type: `byte[]`
Byte array to send to the printer.

Remarks

Check Application.LastError property to see if the method was completed successfully.

See also

- Printer Object

PrintStack Object

A **PrintStack** object represents an opened print job. You can get a reference to a **PrintStack** object by calling the Application.OpenPrintStack method.

Properties

	Name	Description
	FilePath	Gets the path of the opened print stack.

Methods

	Name	Description
	Print ()	Prints the opened print stack.

FilePath Property

Gets the path to the opened print stack. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.50.1010

Syntax

```
PrintStack.FilePath
```

Type

```
String
```

See also

- PrintStack Object

Print Method

Prints the opened print stack.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 6.50.1010

Syntax

```
PrintStack.Print ()
```

Note

Check Application.LastError property to see if the method was completed successfully.

See also

- PrintStack Object

VersionInfo Object

A **VersionInfo** object represents the version information about the API on top of which the application runs. You can get a reference to a **VersionInfo** object through the Application.Info property. The version information is useful to ensure the application is using the proper version of the API.

Properties

	Name	Description
	CompanyName	Gets the company name associated with the API.
	CompiledVersion	Internal version number of the API (this field is for internal use only - see DisplayVersion property for the version string that is displayed to the users).
	Copyright	Gets the copyright notice associated with the API.
	DisplayVersion	Version to be displayed to the users.
	ProductName	Gets the product name associated with the API.

CompanyName Property

Gets the company name associated with the API. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
VersionInfo.CompanyName
```

Type

```
String
```

See also

- VersionInfo Object

CompiledVersion Property

Internal version number of the API (this field is for internal use only - see DisplayVersion property for the version string that is displayed to the users). Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
VersionInfo.CompiledVersion
```

Type

```
String
```

Example

Compiled Version: "4.10.1010"

See also

- VersionInfo Object

Copyright Property

Gets the copyright notice associated with the API. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
VersionInfo.Copyright
```

Type

```
String
```

See also

- VersionInfo Object

DisplayVersion Property

Version to be displayed to the users. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
VersionInfo.DisplayVersion
```

Type

```
String
```

Example

Display Version: "Version 4.10 Build 1010"

See also

- VersionInfo Object

ProductName Property

Name of the product. Read-only property.

Namespace: LSOffice
Assembly: LSOffice.dll
Version: 4.10.1010

Syntax

```
VersionInfo.ProductName
```

Type

```
String
```

See also

- VersionInfo Object

Error Codes and Messages

Labelstar Office follows the standard Automation approach for generating errors.

Runtime Error Handling

When handling errors, determine if the error was caused by using Automation incorrectly, or if the error was returned from an Automation feature API call. When the error was returned from an Automation feature API call use Application.LastError property to get further error information. Each call to an Automation feature API (expect LastError) resets the error information, so that the Application.LastError property obtains error information only for the most recent Automation feature API call.

Runtime Error Codes

Error Code	Description	Type
1000	Windows exception.	Error
1001	The application is alread initialized.	Warning
1002	Invalid license key.	Warning
1003	Invalid license type; Professional license expected.	Warning
1004	Application not initialized; call Initialize() first.	Error
1005	No file opened.	Error
1006	Invalid field name.	Error
1007	No field selected.	Error
1008	Field index out of range.	Error
1009	Invalid property name.	Error
1010	No database fields defined on the label.	Error
1011	No record found.	Error
1012	Multiple database connections.	Error
1013	Invalid printer type; Carl Valentin label printer expected.	Error
1014	Invalid printer name.	Error
1015	No printer selected.	Error
1016	Generic error.	Error
1017	**Labelstar Office** not installed on this computer.	Error
1018	Invalid database name.	Error

Contacts

Product Website

You may find additional information and the latest program version on or website: www.carl-valentin.de

Email

Technical support: support@carl-valentin.de
Ordering and licensing requests: order@carl-valentin.de
General requests: info@valentin-carl.de

System Requirements

Minimal system requirements

- Microsoft Windows 7 SP1/8/8.1/10 x86/x64
- .Net Framework 4.6 or later (download from http://www.microsoft.com/net/)
- Microsoft Visual C++ 2015 Redistributable (x86) (download from https://www.microsoft.com/en-us/download/details.aspx?id=53587)
- Microsoft Access Database Engine 2010 (x86)
- Recommended printer drivers: Carl Valentin Printer Drivers Version 2.4.1 or later

Note

Some components, such as .Net Framework for example, are not included in the installation program by default. During installation, the program searches for components, downloads them from the internet where applicable and installs them. If you do not have internet access, you can find the necessary components on the program CD in the *Utilities* folder.

Imprint

Carl Valentin GmbH
Neckarstrasse 78-86 u. 94
78056 Villingen-Schwenningen

Phone: +49 (0) 7720 9712 - 0
Email: info@carl-valentin.de

Copyright © 2019 Carl Valentin GmbH

Disclaimer

These descriptions however do not constitute any guaranteed characteristics in a legal sense or in the sense of any product liability. The authors reserve the right to make changes to the software, to announce any person without obligation these changes. No warranty is accepted for the correctness of the contents of this document. As errors can never be completely avoided, despite all efforts made, we are grateful for any information regarding errors.

Trademark Notifications

All product names mentioned in this document may be trademarks or registered trademarks of their respective owners.