the professional labeling software

www.nicelabel.com, info@nicelabel.com

NiceLabel Programming Guide

English Edition

Version 20110830-27

Head Office
Euro Plus d.o.o.
Poslovna cona A 2
SI-4208 Šenčur
Slovenia
Tel: +386 4 280 50 00
Fax: +386 4 233 11 48
sales@nicelabel.com
http://www.nicelabel.com

Index

1. Introduction .... 1—5

1.1 Overview of software integration .... 1—5

1.2 Introduction to DDE and ActiveX .... 1—6

1.2.1 What is Dynamic Data Exchange (DDE)? .... 1—6
1.2.2 What is COM / ActiveX? .... 1—6
1.2.3 What is a Type Library? .... 1—7
1.2.4 The OLE/COM Object Viewer .... 1—8

1.3 What is Visual Basic .... 1—8

1.4 What is Visual Studio .NET .... 1—8

1.5 What is ASP? .... 1—9

2. ActiveX interface .... 2—10

2.1 Introduction .... 2—10

2.2 How to use NiceLabel ActiveX interface .... 2—10

2.2.1 Basic Method .... 2—10
2.2.2 Advanced Method .... 2—11
2.2.3 Differences between early and late binding .... 2—11

2.3 NiceLabel Wrapper .... 2—14

2.3.1 What is NiceLabel Wrapper? .... 2—14
2.3.2 NiceLabel Wrapper Features: .... 2—14
2.3.3 NiceLabel Wrapper Functionality .... 2—14
2.3.4 Usage of NiceLabel Wrapper .... 2—14
2.3.5 Where to get NiceLabel Wrapper? .... 2—15

2.4 NiceLabel Engine Wrapper .... 2—15

2.4.1 What is NiceLabel Engine Wrapper? .... 2—15
2.4.2 NiceLabel Engine Wrapper Features: .... 2—15
2.4.3 NiceLabel Engine Wrapper Functionality .... 2—15
2.4.4 Usage of NiceLabel Engine Wrapper .... 2—16
2.4.5 Where to get NiceLabel Engine Wrapper? .... 2—16

2.5 Create instance of the installed NiceLabel .... 2—16

2.6 Functional comparison between NiceLabel Engine version 5 and NiceLabel Pro version 5 .... 2—17

2.7 Logging support for NiceLabel SDK version 5 .... 2—17

2.8 NiceLabel SDK Unit Based licensing .... 2—17

2.8.1 Sofware key licensing .... 2—17

2.9 NiceLabel ActiveX common interfaces .... 2—21

2.9.1 Class Application .... 2—21
2.9.2 Hierarchy diagram .... 2—37
2.9.3 Class IBarcode (Advanced only) .... 2—38

2.9.4 Class IDatabase (Advanced Only) .... 2—52
2.9.5 Class IDatabaseList (Advanced Only) .... 2—59
2.9.6 Class IDBDef .... 2—61
2.9.7 Class IDBField .... 2—62
2.9.8 Class IDBParameter (Advanced Only) .... 2—64
2.9.9 Class IEXTFunction (Advanced Only) .... 2—66
2.9.10 Class IFieldList (Advanced Only) .... 2—69
2.9.11 Class IFunction (Advanced Only) .... 2—70
2.9.12 Class IFunctionList (Advanced Only) .... 2—73
2.9.13 Class IGraphics (Advanced Only) .... 2—74
2.9.14 Class ILabel (Advanced Only) .... 2—85
2.9.15 Class INiceLabel (Advanced Only) .... 2—104
2.9.16 Class IObject (Advanced Only) .... 2—128
2.9.17 Class IObjectList (Advanced Only) .... 2—137
2.9.18 Class IOleObject (Advanced Only) .... 2—137
2.9.19 Class IParagraph (Advanced Only) .... 2—146
2.9.20 Class IParameterList (Advanced Only) .... 2—162
2.9.21 Class IRectangle (Advanced Only) .... 2—162
2.9.22 Class IRTFText (Advanced Only) .... 2—174
2.9.23 Class IText (Advanced Only) .... 2—189
2.9.24 Class IVar (Advanced Only) .... 2—206
2.9.25 Class IVariableList (Advanced Only) .... 2—224
2.9.26 Class ILabelSettings (Advanced Only) .... 2—229

2.10 NiceLabel ActiveX interfaces for advanced users .... 2—244
2.10.1 Class Application .... 2—244
2.10.2 Class IDatabase .... 2—247
2.10.3 Class IEXTFunction (Advanced Only) .... 2—248
2.10.4 Class IFunction .... 2—249
2.10.5 Class ILabel .... 2—251
2.10.6 Class IParagraph .... 2—252
2.10.7 Class IRTFText .... 2—252
2.10.8 Class IVar .... 2—253

2.11 Advanced printing via ActiveX interface .... 2—254
2.11.1 How to use Nlog5 Interface .... 2—254
2.11.2 Class IPrintJobFactory .... 2—255

2.12 NiceForm ActiveX Interfaces .... 2—256
2.12.1 Introduction .... 2—256
2.12.2 NiceForm ActiveX Interface - Design mode .... 2—257
2.12.3 NiceForm ActiveX Interface - Run mode .... 2—258

3. LabelServices Interface .... 3—260

3.1 Introduction .... 3—260

3.2 Class TPQPrinter .... 3—260

3.3 Class TPQPrinterList .... 3—263

3.4 Class TPQPrinterJob .... 3—263

3.5 Class TPQPrinterJobList .... 3—267

3.6 Types .... 3—267

3.7 Methods ...3—269

4. DDE Communication ...4—278
4.1 Introduction ...4—278
4.2 DDE Commands ...4—278
4.2.1 DDEInitiate ...4—278
4.2.2 DDEExecute ...4—278
4.2.3 DDETerminate ...4—278
4.3 NiceCommands ...4—279
4.3.1 COMMON ...4—279
4.3.2 FILE ...4—282
4.3.3 PRINTING ...4—283
4.3.4 DATABASE ...4—286

5. Programming Samples ...5—288
5.1 DDE ...5—288
5.2 ActiveX interface ...5—288
5.3 ActiveX Hints ...5—288
5.4 Active Server Pages ...5—291
5.5 ASP.NET ...5—293
5.6 Visual Basic Script (VBScript) ...5—294

6. FAQ ...6—295
6.1 How to choose different printer for label printing not the one than is selected in the label? ...6—295
6.2 How to close the label, if you do not have Label ID? ...6—295
6.3 How to print label, if you do not have Label ID? ...6—295
6.4 How to save the label with NiceLabel API? ...6—295
6.5 How can I put NiceLabel in the foreground when I control it with ActiveX interface? ...6—296
6.6 How can I use NiceLabel API in .NET environment? ...6—296
6.7 Where can I get programming samples for different environments? ...6—296
6.8 Can I Control NiceForm through the ActiveX Interface? ...6—296

7. Glossary ...7—297

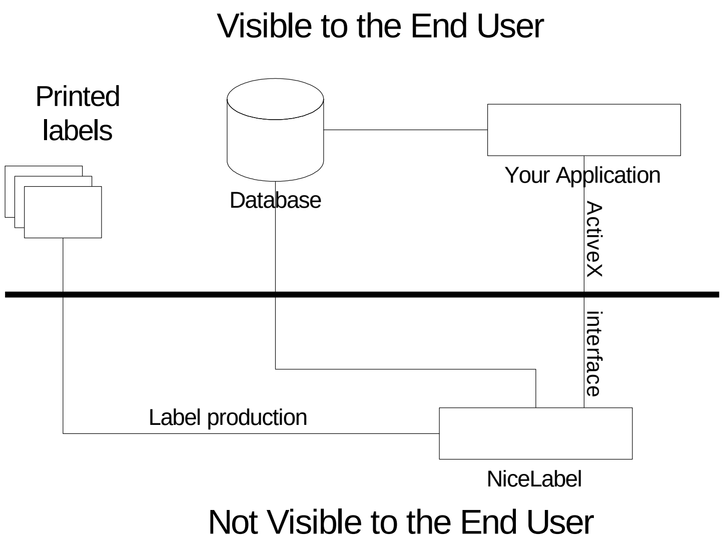
8. Contacts ...8—299

1. Introduction

1.1 Overview of software integration

The most common method of label print is directly from NiceLabel software using its GUI (graphic user interface). But sometimes there might be other requirements for label production. NiceLabel has a wide variety of connectivity and integration options so you do not need to use NiceLabel interactively, but through an ActiveX interface or Dynamic Data Exchange connectivity.

NiceLabel can be used as a "print-engine" totally integrated to your custom application and invisible to the end-user eyes.



Basically, the end user sees only your custom application that connects to NiceLabel and uses NiceLabel label printing power in the background.

The purpose of this manual is to show you how you can control the NiceLabel software from your own application that will be deployed to the end-user. Everything you must know about integration of NiceLabel into your application is described in this manual.

The information in this manual is for advanced users and application developers only. If you do not plan to write applications that use integrated NiceLabel to print labels, you can skip reading this manual entirely. If you are interested in NiceLabel connectivity and integration options in general, please refer to the White Paper covering these options. You can download the White Paper from NiceLabel website at www.nicelabel.com.

All methods and properties of NiceLabel ActiveX object described in this document are available in **NiceLabel Pro** application. NiceLabel Pro application is available in many NiceLabel editions, like NiceLabel Pro, NiceLabel Pro Print-Only, NiceLabel Suite, NiceLabel Suite Print-Only, NiceLabel Suite Network, NiceLabel SDK and other.

NiceLabel Express and NiceLabel SE editions do not have the ActiveX interface.

This manual contains a FAQ section where most frequently asked questions are answered.

1.2 Introduction to DDE and ActiveX

1.2.1 What is Dynamic Data Exchange (DDE)?

Microsoft Windows allows multiple programs to run at the same time. Part of power of the Windows environment lies in the ability of these programs to communicate with each other. Various methods of inter-process communication are available in Windows, including the "clipboard," Object Linking and Embedding (OLE), and Dynamic Data Exchange (DDE).

The Dynamic Data Exchange mechanism is one way for software applications to communicate with each other. This communication has two forms: data and commands. Data are identified by the application name and by individual topic and item names within that application. It is also possible to create hot links, so that one application can notify another when information has changed. Applications can also execute commands in other applications.

From the user's perspective, *establishing* a DDE link is often no different from cutting and pasting non-dynamic data between applications. The user can often establish a DDE link by "copying" the data from one application, and then, in a second application choosing a command like "Paste Link..." from a menu. The DDE link then continues to exist without further action on the user's part.

The developer's view of DDE is quite different. At its most basic level, DDE is simply a message-based protocol, which applications can use to share data in global memory. Applications can share data on a one-time basis or on an ongoing basis in which new data is automatically sent from one application to another when it becomes available.

Dynamic Data Exchange or DDE is a Windows feature that allows Windows applications to communicate with each other. DDE is based on the messaging system built into Windows. Two Windows programs can carry on a DDE "conversation" by posting messages to each other. These two programs are known as the "server" and the "client". A DDE server is the program that has access to data that may be useful to other Windows programs. A DDE client is the program that obtains this data from the server.

1.2.2 What is COM / ActiveX?

The Component Object Model (COM) is a binary standard designed to support reusable, language-independent and multi-platform components. With COM, it is possible for one component to communicate with another component. These components can be on different threads, processes, or even machines.

ActiveX is Microsoft’s implementation of COM. COM and ActiveX are often used interchangeably in different technical documentation. An *ActiveX control* is just a special user interface ActiveX (COM) object designed for use on forms.

COM objects define *interfaces* that expose methods and properties by which clients (user applications) can manipulate the object. In fact, objects can only be manipulated via an interface. Interfaces are immutable, which means that once an interface has been published, it can never change. In other words, methods should never be added, removed, or have their signatures changed. A new version of the object could be provided by a new interface to expose any new functionality while keeping the original interface intact. Objects typically have many interfaces. A method of one interface may access, use, or return another interface.

1.2.3 What is a Type Library?

A Type Library is a file that contains descriptions of a component's classes, interfaces, data types, and methods. The type library may be a stand-alone file (usually with an **.OLB** extension) or may be embedded within the runtime file (DLL or executable). The contents of a type library may be inspected with a type library browser such as **OLEVIEW.EXE** that can be downloaded from the Microsoft web site.

The type library supplied by the NiceLabel labeling software is a file called **Nlabel5.OLB** that describes the type of all of the ActiveX objects. For NiceLabel version 4 suitable file is NICE4.OLB , for NiceLabel version 3 suitable file is NICE3.OLB

The type library does not store objects, but it stores type information. By accessing the type library, your application can determine the characteristics of an object, such as the interfaces supported by the object and its names and parameters. This library helps you to write your program because it contains all of definitions of object methods and properties that you can access. Using the type library you can optimize your job.

The name of the type library is **NiceLabel**.

The following procedures show how to install and use the type library with Visual Basic 6.0 , Delphi.

<u>Visual Basic</u>

To install the type library:

- Choose Project -> References.
- Activate NiceLabel in the list of available references and validate the dialog box.

To display the methods and properties:

- Use the Object Explorer
- In the library list select NiceLabel

You can run **Object Explorer** anytime with pressing **F2** button on your keyboard.

To use the type library

While writing code, you have just to enter a period after an object to get the associated methods and properties, or after a method to get the associated properties.

<u>Delphi</u>

- Choose Project -> Import Type Library
- Activate NiceLabel in the list of available references and validate the dialog box.

1.2.4 The OLE/COM Object Viewer

The OLE/COM Object Viewer is a developer- and power user-oriented administration and testing tool. With the OLE/COM Object Viewer you can view the NiceLabel type library contents. Use the Viewer to figure out what methods, properties, and events NiceLabel supports.

To create the NiceLabel type library go to the command prompt and type the following command (you have to be positioned in the folder where NiceLabel main file NICE4.EXE is stored) For version 3 Nice3.exe file must be executed.

Valid for version 5:

NLABEL5.EXE /typelib

Valid for version 4:

```
NICE4.EXE /typelib
```

Valid for version 3:

```
NICE3.EXE /typelib
```

This produces the file **NICELABEL5.OLB**. You can view this file with the OLE/COM Object Viewer.

OLE/COM Object Viewer is available to download from:

http://www.microsoft.com/com/resources/oleview.asp

1.3 What is Visual Basic

Visual Basic (VB) is a RAD (Rapid Application Development) tool that enables programmers to create Windows applications in a very short period of time. It is the most popular programming language in the world, and has more programmers and lines of code than any other competitive development language.

1.4 What is Visual Studio .NET

Visual Studio .NET is the comprehensive, multi-language development tool for rapidly building and integrating applications. Visual Studio .NET offers a highly productive environment in which to develop a broad range of Microsoft .NET connected applications and technologies. Using the high-performance Microsoft .NET Framework run-time environment, Visual Studio .NET provides you with powerful tools for designing, building, testing, and deploying applications.

1.5 What is ASP?

Active Server Pages are HTML pages that contain embedded scripts. IIS (Internet Information Server) and third party providers offer server software that interprets Active Server code and displays the result in the user's internet browser application.

ASP pages contain either server side or client side scripts, which performs functions such as database access, page personalization, or interactive functions.

On In the same way the NiceLabel labeling software can be controlled.

For more technical details and samples please refer to the chapter **Active Server Pages** on the page 5—291

2. ActiveX interface

2.1 Introduction

ActiveX is a technology used within Windows that allows the programs to interact with one another through a pre-defined interface. ActiveX was previously known as OLE Automation and is still referred to as such in the documentation of many programming languages that can make use of it.

An ActiveX interface is made up of 'exposed objects and methods'. The objects can be looked upon as the containers for the methods that provide the interface functionality. The methods perform specific tasks using parameters you pass to the method to specify the detail of the task.

ActiveX interface available in NiceLabel software allows you to take remote control over label printing. You can embed the functionality of NiceLabel Pro into your application and use NiceLabel as print engine.

Before you start developing your custom application you should know, how to use NiceLabel ActiveX interface in your software.

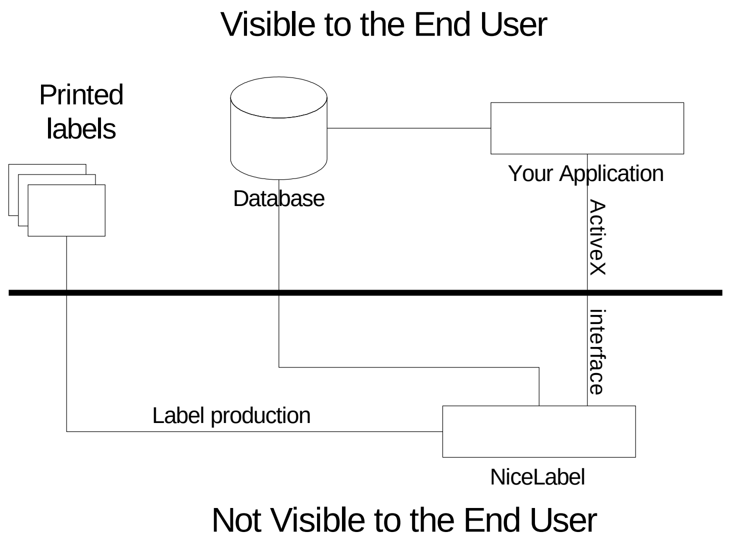


Figure 1: Schematic view on communication paths when using NiceLabel ActiveX interface

2.2 How to use NiceLabel ActiveX interface

Integration of NiceLabel can be done using two methods. Each of them has different approach for accessing NiceLabel through ActiveX and must be understood, otherwise you can experience troubles with the ActiveX interface. The most important thing to remember is that you can use only one method at a time.

The two methods must never be mixed so you must decide which method you will use for the integration and then stick with it.

2.2.1 Basic Method

The basic method is suitable for simple operations such as:

- Opening labels
- Setting variables
- Setting printers
- Printing

The basic connection to the label must be done with the method ***LabelOpen***. When accessing NiceLabel with LabelOpen command, you will get the **ID** of the label. All further operations that are done with the label are then managed with this ID.

The basic method allows you to use the basic operations, but they are quite sufficient for label printing. It is not possible to use any of the label's interfaces, their properties and/or methods as with the advanced method. If you need more power over your label, you have the advanced method available.
This method does not support Unicode characters.

2.2.2 Advanced Method

The advanced method is suitable for simple and also advanced operations such as:

- Opening labels
- Setting variables
- Setting printers
- Printing
- **Getting label preview**
- **Changing the label design on-the-fly**

The advanced connection to the label must be done with the ***LabelOpenEx*** command. When accessing NiceLabel with LabelOpenEx command then you get the **interface** to the label. Label interface can be used to access all available NiceLabel ActiveX interfaces and their properties and/or methods described in this manual

Please refer to the figure 2 for the hierarchy of the interfaces.

IMPORTANT!

The commands LabelOpen and LabelOpenEx are not equivalents and **must not be mixed**. Use only one method at a time. When using LabelOpen, you interact with the label using label ID, when using LabelOpenEx, you interact with the label base on the label interface.

2.2.3 Differences between early and late binding

There are two method for using ActiveX interface to control NiceLabel, late binding and early binding.

<u>Late binding</u>

Late binding uses the CreateObject command to create an instance of the NiceLabel object that you then control from your application. For example, to create a new instance of NiceLabel using late binding you would use the command:

```
Dim oNice As Object
Set oNice = CreateObject("NiceLabel5.Application")
```

Valid for NiceLabel 3

```
Dim oNice As Object
Set oNice = CreateObject("NiceLabel.Application")
```

Valid for NiceLabel 4

```
Dim oNice As Object
Set oNice = CreateObject("NiceLabel4.Application")
```

Valid for NiceLabel 5

```
Dim oNice As Object
Set oNice = CreateObject("NiceLabel5.Application")
```

Early binding

To use early binding, you must first set a reference to NiceLabel ActiveX interface. The reference is stored in the type library file. For more information refer to the chapter **What is a Type Library?**

If you need to integrate NiceLabel print engine to application running in .NET environment, you will have to add NiceLabel Wrapper DLL to your development environment in the same ways as the type library.

To create a new instance of NiceLabel using early binding use the command:

```
Dim oNice As NiceApp
Set oNice = New NiceApp
```

NiceApp class is part of **NiceLabel Wrapper**. More about NiceLabel Wrapper please read in its section in this document.

In either case, you can first try to get an existing instance of NiceLabel. If NiceLabel is not already running, the action will return an error message, so you will have to run a new instance of NiceLabel.

To connect to an already running instance of NiceLabel you would use GetObject (regardless whether you're using early or late binding). If existing NiceLabel is not found in memory, the error handler will be returned for your evaluation.

```
Dim oXL As Object
Set oXL = GetObject("NiceLabel5.Application")
```

Valid for NiceLabel 3

```
Dim oXL As Object
Set oXL = GetObject("NiceLabel.Application")
```

Valid for NiceLabel 4

```
Dim oXL As Object
Set oXL = GetObject("NiceLabel4.Application")
```

Valid for NiceLabel 5

```
Dim oXL As Object
Set oXL = GetObject("NiceLabel5.Application")
```

Advantages of Early Binding vs. Late Binding

When using early binding, your code will run considerably faster, because it can all be compiled up front. With late binding, the code relating to an application you declared as an object has to be compiled as it runs. It slows the overall speed of your application.

Because your code can all be compiled up front, debugging of the code is far easier. You need to select Debug + Compile option in your development environment and the compiler will be able to spot syntax errors which would have been missed had you used late binding.

When using early binding in your project you also have the full access to intellisense. IntelliSense provides an array of options that make language references easily accessible. When coding, you do not need to leave the Code Editor or the Immediate Mode command window to perform searches on the language elements. You can keep your context, find the information you need, insert language elements directly into your code, and even have IntelliSense complete your typing for you.

With intellisense you can enter the program code faster than without it. You can type a keyword followed by a dot to get a popup list of properties and methods supported by that keyword. The access to the help for specified keyword is also simplified with early binding. When you type a keyword and press F1 the online help topic for that keyword is automatically displayed. You have the full access to the application's object model via the Object Browser and VBA Help.

Additionally, you have access to the application's built-in constants. Furthermore, while typing a source code for an object, you will get a pop-up list of the supported methods or properties.

All these advantages make programming using early binding immeasurably easier than using late binding.

2.3 NiceLabel Wrapper

2.3.1 What is NiceLabel Wrapper?

NiceLabel Wrapper is a dynamic link library that helps you integrate NiceLabel ActiveX interface in .NET and VB Applications better and with more power. NiceLabel Wrapper has some additional approaches that are not present in the NiceLabel ActiveX interface alone.

2.3.2 NiceLabel Wrapper Features:

- Early binding available
- Possible Usage in .NET environment

2.3.3 NiceLabel Wrapper Functionality

NiceLabel Wrapper is the library you interact with from your application. Using the Wrapper you no longer connect directly to NiceLabel ActiveX interface, but leave this job to the Wrapper.

NiceLabel Wrapper is practically completely transparent with NiceLabel ActiveX. All interfaces, methods and properties available in the NiceLabel Wrapper have the same name and functionality as in the original NiceLabel ActiveX interface.

There are only two differences.

1. Additional “Free” methods in all classes (except NiceApp class, which already had Quit method). These methods should be used in .NET environment to ensure that no longer needed NiceLabel objects are correctly released from the memory.
2. Some class names are different:
   WRVar instead of Var
   WRObject instead of Object
   WRFunction instead of Function

2.3.4 Usage of NiceLabel Wrapper

Visual Basic

You must include Nicelabel wrapper in your project:

- Choose Project -> References.
- Activate NiceLabel5WR (Nicelabel4WR) in the list of available references and validate the dialog box.

.After that you have the possibility to use NiceLabel ActiveX types (NiceApp, NiceLabel, etc), methods and properties from your application. Early binding is also available. Visual Basic 6 has no access to classes start with I, like IBarcode, IText,.. With usage of Type Library (chapter 1.2.3) Visual Basic 6 has access to classes start with I.

.Net – Microsoft Visual Studio

You must include Nicelabel wrapper in your project:

- Choose Project -> Add References.
- Activate NiceLabel5WR (Nicelabel4WR) in the list of available references in .COM tab

After that use of NiceLabel ActiveX types (NiceApp, NiceLabel, etc), methods and properties is possible (early binding). After the certain object is no longer needed you must use the proper “Free” method to ensure that no longer needed NiceLabel objects are released correctly (otherwise garbage collector may not dispose them immediately).

2.3.5 Where to get NiceLabel Wrapper?

NiceLabel Wrapper version 5 dll file is installed together with the NiceLabel itself. Default location of the file is C:\Program Files\EuroPlus\NiceLabel 5\Bin

NiceLabel Wrapper version 4 dll file is installed together with the NiceLabel itself. Default location of the file is C:\Program Files\EuroPlus\NiceLabel 4\Bin

2.4 NiceLabel Engine Wrapper

2.4.1 What is NiceLabel Engine Wrapper?

NiceLabel Engine Wrapper is a dynamic link library that helps you integrate NiceLabel Engine interface in .NET and VB Applications better and with more power. NiceLabel Engine Wrapper has some additional approaches that are not present in the NiceLabel Engine interface alone.

2.4.2 NiceLabel Engine Wrapper Features:

- Early binding available
- Possible Usage in .NET environment

2.4.3 NiceLabel Engine Wrapper Functionality

NiceLabel Engine Wrapper is the library you interact with from your application. Using the Wrapper you no longer connect directly to NiceLabel Engine interface, but leave this job to the Wrapper. The Wrapper runs between your application and the NiceLabel Engine.

NiceLabel Engine is practically completely transparent with NiceLabel Engine. All interfaces, methods and properties available in the NiceLabel Engine Wrapper have the same name and functionality as in the original NiceLabel Engine interface.

2.4.4 Usage of NiceLabel Engine Wrapper

Visual Basic

You must include NiceEngine wrapper in your project:

- Choose Project -> References.
- Activate NiceEngine5WR (NiceEngine4WR) in the list of available references and validate the dialog box.

After that you have the possibility to use NiceEngine ActiveX types (NiceApp, NiceLabel, etc), methods and properties from your application. Early binding is also available.

.Net – Microsoft Visual Studio

You must include NiceEngine wrapper in your project:

- Choose Project -> Add References.
- Activate NiceEngine5WR (NiceEngine4WR) in the list of available references in .COM tab

After that use of NiceEngine ActiveX types (NiceApp, NiceLabel, etc), methods and properties is possible (early binding). After the certain object is no longer needed you must use the proper "Free" method to ensure that no longer needed NiceLabel objects are released correctly (otherwise garbage collector may not dispose them immediately).

2.4.5 Where to get NiceLabel Engine Wrapper?

NiceEngine Wrapper version 5 dll file is installed together with the NiceLabel itself. Default location of the file is c:\Program Files\Common Files\EuroPlus Shared\NiceEngine 5

NiceEngine Wrapper version 4 dll file is installed together with the NiceLabel itself. Default location of the file is C:\Program Files\EuroPlus\NiceLabel 4\Bin

2.5 Create instance of the installed NiceLabel

NiceLabel ActiveX interface supports version 3, version 4 and version 5. Version 5 ActiveX Interface includes all methods and properties described below. Previous versions (3 and 4) do not include all methods and properties.

To connect to an installed of NiceLabel it is advised to use following code.
If existing NiceLabel version is not installed, object variable reference is nothing and another version of the NiceLabel has to be choosing. Check Version Availability for each method or property.

```
Dim oXL As Object
Set oXL = CreateObject("NiceLabel5.Application")
If oXL is Nothing then Set oXL = CreateObject("NiceLabel4.Application")
  If oXL is Nothing then Set oXL = CreateObject("NiceLabel3.Application")
```

2.6 Functional comparison between NiceLabel Engine version 5 and NiceLabel Pro version 5

	NiceLabel Engine	NiceLabel Pro
Create labels	NO	YES
Print labels	YES	YES
Needs custom application for initiating of label printing?	YES	NO
ActiveX programming interface	YES	YES
Instance quantity	MORE	ONE
Class Name	NiceLabel5.Engine	NiceLabel5.Application

2.7 Logging support for NiceLabel SDK version 5

NiceLabel SDK version 5 has built-in support for logging of label printing operations. Whenever a label print even occurs all information about the events are stored in the log file. By default logging functionality is disabled.
If you want to log the information about every printed label, you just need to enable the logging functionality.

For more information refer to the **Nicelabel SDK User Guide.**

2.8 NiceLabel SDK Unit Based licensing

NiceLabel SDK Unit Based can licensing by:

- Hardware key
- Software key

For hardware key licensing refer to **Nicelabel SDK User Guide.**

2.8.1 Sofware key licensing

Following options are available:

- During product (third party application) installation, user could enter the product key number, and the install program can call the procedure to activate the license on-line

- During product uninstallation, the install program can call the procedure to de-activate license on-line
- When on-line internet connection is not available, product's installation can call a routine, that will start the Activation Wizard. Using the SDK's activation WebPage, product can be fully activated using Registration Number and Activation Code.
- Prior to calling the on-line activation routine, install program can verify the entered key number (just for validity of the key)
- Status of the activation can also be verified (if the product is already installed or not)

SDKLicense.dll needs for Software key licensing. It is installed together with NiceLabel SDK deployment package. Third party application can call this DLL to achieve activation/deactivation of the SDK package.
The DLL is located in the location, where NiceEngine is installed. This is normally on C:\Program Files\Common Files\EuroPlus Shared\NiceEngine 5

Function	Version Availability
SLInit	5.2
SLActivateLicense	5.2
SLDeactivateLicense	5.2
SLGetLicense	5.2
SLManageLicense	5.2
SLValidateKeyNumber	5.2

SLInit

Description:
Function must be called prior to all other functions. It sets the language of the library, error messages will be returned in desired languages.

Possible values for the languages:

English	= 1
Slovenian	= 2
Croatian	= 3
German	= 4
Danish	= 5
French	= 6
Spanish	= 7
Italian	= 8
Finish	= 9
Dutch	= 10
Portuguese	= 11
Czech	= 12
Polish	= 13
Hungarian	= 14
Russian	= 15
Greek	= 16
Swedish	= 18

Hebrew	= 19
Slovak	= 20
Lithuanian	= 21
Chinese (Traditional)	= 22
Chinese (Simplified)	= 23
Thai	= 24
Korean	= 25
Japanese	= 26
Ukrainian	= 27
Norwegian	= 29

Return Value is 0.

Syntax:

```
Function SLInit (Language: integer) : integer
```

SLActivateLicense

Description:
Function will perform on-line activation of the key number provided in the KeyNumber parameter. To perform the activation, Name, Company, Country and Email parameters must be provided.
Reason parameter needs to be provided, if the key number is activated not for the first time. See return codes for more explanation.
In case of an error, the buffer provided by ErrorMessage parameter is filled with the message, up to MaxMsgLen length.

Return value of the call:
0 – success
2 – Reason needs to be provided to the call
3 – Country was not provided
4 – Email was not provided
Other non-zero value – unknown error happened. Detailed description of the error is provided in ErrorMessage parameter

Syntax:

```
Function SLActivateLicense (Name, Company, Country, EMail, Reason,
KeyNumber: PChar; ErrorMessage: PChar; MaxMsgLen: integer) :
integer
```

SLDeactivateLicense

Description:
Function will perform on-line deactivation of the key number provided in the KeyNumber parameter.
In case of an error, the buffer provided by ErrorMessage parameter is filled with the message, up to MaxMsgLen length.

Return value of the call:
0 – success
-1 – Error happened. Detailed description of the error is provided in ErrorMessage parameter

Syntax:

```
Function SLDeactivateLicense (KeyNumber: PChar; ErrorMessage:
PChar; MaxMsgLen: integer)
```

SLGetLicense

Description:
Function verifies the status of activation on the system. When the product is already activated, the KeyNumber is returned in KeyNumber parameter (buffer of 26 or more bytes needs to be provided.

When SolutionID parameter is provided, it will be filled with the SolutionID parameter of the Key (refer to **Nicelabel SDK User Guide** for more information)

Return value:
0 – product is activated
-1 – product is not activated

Syntax:

```
Function SLGetLicense (KeyNumber: PChar; SolutionID: PWORD):
integer
```

SLManageLicense

Description:
Function is called to perform activation of the product (product is not activated yet) or show the license information (when product is already activated).
Function will show the activation wizard, which will guide the user through the process of license activation.

Return value:
0 – success

Syntax:

```
function SLManageLicense (hParent: THandle) : integer
```

SLValidateKeyNumber

Description:
Function verifies the key, provided in KeyNumber parameter.

Return value:
0 – Key is valid
-1 – Key is not valid.

Syntax:

```
Function SLValidateKeyNumber (KeyNumber: PChar) : integer
```

2.9 NiceLabel ActiveX common interfaces

This section contains description of the common methods and properties.

2.9.1 Class Application

Methods	Version Availability	Properties	Version Availability
ExecuteMacro	3,4,5	Application	3,4,5
GetOpenedLabels	4	DetailedMessage	3,4,5
GetVersion	4,5	ErrorID	3,4,5
GetLabelDownloadVariablesList	5	ErrorMessage	3,4,5
IsLabelExportable	3,4,5	IsDemo	4,5
JobRun	3,4,5	ProcessID	4,5
LabelClose	3,4,5		
LabelCreate	3,4,5		
LabelExportToPocket	3,4,5		
LabelGeneralExport	3,4,5		
LabelGetVarCount	3,4,5		
LabelGetVarName	3,4,5		
LabelGetVarProperty	3,4,5		
LabelOpen	3,4,5		
LabelOpenEx	3,4,5		
LabelOpenExForPrinter	4,5		
LabelOpenForPrinter	4,5		
LabelPrint	3,4,5		
LabelSessionEnd	3,4,5		
LabelSessionPrint	3,4,5		
LabelSessionStart	3,4,5		
LabelSetPrinter	3,4,5		
LabelSetPrintJobName	4,5		
LabelSetVar	3,4,5		

LabelTestConnection	3,4,5		
Login	3,4,5		
Quit	3,4,5		
SetApplicationProperty	5		
SetDefaultPrinter	4,5		

Application

Description:

Application method returns an interface to the niceApp.

Syntax:

```
Property Application As NiceLabel
```

Access Rights:

```
read-only
```

Method:

Advanced

DetailedMessage

Description:

This is a read-only property, which holds additional information (if they exist) about the error (if it has occurred).

Syntax:

```
Property DetailedMessage As String
```

Access Rights:

read-only

Method:

Basic, Advanced

See also:

ErrorID, ErrorMessage

ErrorID

Description:

This is a read-only property that contains last error code (if error occurred).

Syntax:

```
Property ErrorID As Long
```

Access Rights:

read-only

Method:

Basic, Advanced

See also:

DetailedMessage, ErrorMessage

ErrorMessage

Description:

This is a read-only property that holds the last error message (if the error has occurred).

Syntax:

```
Property ErrorMessage As String
```

Access Rights:

read-only

Method:

Basic, Advanced

See also:

DetailedMessage, ErrorID

ExecuteMacro

Description:

This function executes a Macro (DDE) command.

Syntax;

```
Function ExecuteMacro(LabelID As Long, Macro As String) As Boolean
```

Method:

```
Basic
```

GetOpenedLabels

Description:

The niceApp interface is extended to support returning a list of opened labels. The command is defined as:

Method returns a multiline text string, each line containing the path name of the opened label. In case the label is not saved yet (path name does not exist), the title of the label is returned.

Syntax:

```
GetOpenedLabels()
```

Method:

Advanced

See also:

LabelOpenEx

GetVersion

Description:

GetVersion method can be used to check NiceLabel version.
Method returns a string value regarding to VersionID value.
VersionID value can be of the following:
0 – complete info
1 - Major version
2 - Minor version
3 - Release version
4 - Build number

Syntax:

```
Function GetVersion(VersionID As Integer) As String
```

Method:

Advanced

GetLabelDownloadVariablesList

Description:

GetLabelDownloadVariablesList method returns list of label variables. It has to be set before using metods like ExportToPrinter or DownloadToPrinter.
Method returns interface to the variable list of label variables.

Syntax:

```
Function GetLabelDownloadVariablesList(LabelID As Long) As
VariableList
```

Method:

Advanced

See also:

ExportToPrinter, DownloadToPrinter

<u>IsDemo</u>

Description:

This property tells the client application whether the server is running in DEMO mode or not.

Syntax:

```
Property IsDemo As Boolean
```

Access Rights:

read-only

Method:

Basic, Advanced

<u>IsLabelExportable</u>

Description:

IsLabelExportable method can be used to check in advance, if the label can be exported. This function can be used from applications, which later use the “ExportToPocketPC” ActiveX method.

When label can be exported, the method returns TRUE.

Syntax:

```
Function IsLabelExportable() As Boolean
```

Method:

Advanced

See also:

LabelExpotToPocket

JobRun

Description:

Function executes specified JOB file.

Syntax:

```
Function JobRun(FileName As String) As Boolean
```

Method:

```
Basic, Advanced
```

LabelClose

Description:

Closes the label with specified ID. If it succeeds TRUE is returned.

Syntax:

```
Function LabelClose(LabelID As Long) As Boolean
```

Method:

```
Basic
```

See also:

LabelOpen

LabelCreate

Description:
Creates new label document and returns interface as a result

Syntax:
```
Function LabelCreate() As INiceLabel
```

Method:

```
Advanced
```

LabelExportToPocket

Description:

Function executes the same action as choosing the command **File|Export|Export to terminal**. All the necessary files are created in directory specified as second parameter.

Syntax:

```
Function LabelExportToPocket(LabelID As Long, ExportDir As String)
As Boolean
```

Method:

```
Basic, Advanced
```

See also:

IsLabelExportable

LabelGeneralExport

Description:

Performs an export of the label specified by LabelID (this label has to be open with LabelOpen call where you retrieve the ID of the label), the exported files are placed into the ExportDir folder

Syntax:

```
Function LabelGeneralExport(LabelID As Long, ExportDir As String)
As Boolean
```

Method:

```
Basic
```

LabelGetVarCount

Description:

Function returns number of variables on the label with specified label ID The returned value is zero based index. To get an actual number of variables on the label you have to add up 1.

Syntax:

```
Function LabelGetVarCount(LabelID As Long) As Long
```

Example:

Label has 5 variables. LabelGetVarCount will return number 4.

Label has 17 variables. LabelGetVarCount will return number 16.

Method:

Basic

See also:

LabelGetVarName, LabelGetVarProperty, LabelSetVar

LabelGetVarName

Description:

Function returns the variable name with the specified variable number (VarID) on the label with specified label ID. VarID is a number of variable from 1 to number of variables on the label (returned by LabelGetVarCount function).

Syntax:

```
Function LabelGetVarName(LabelID As Long, VarID As Long) As String
```

Method:

```
Basic
```

See also:

LabelGetVaCount, LabelGetVarProperty, LabelSetVar

LabelGetVarProperty

Description:

Function returns a property of a variable with the name VarName. PropName specifies name of the property to be returned.

Possible values for PropName are:

Value
Name
ID
Used
VarQuantity
InputType
Description
Prefix
Suffix
Length
FixedLength
MultiLine
LineCount
LineLength
FormatID
Justification
PadChar
WordWrap
MinValue
MaxValue
HasMinValue
HasMaxValue

InputPicture
OutputPicture
PictureType

InputType has the following values:

Value	Description
1	Prompt
2	System clock
3	PrinterClock
4	Global
5	Generated
6	Database

For prompted variables:

"Prompt"

"PromptType"

"DefValue"

"DefType"

"IsDynamic"

"IsRequired"

Variable types are self-explanatory. If the type is boolean, then the return value is either TRUE or FALSE. Otherwise the value is string or number.

Syntax:

```
Function LabelGetVarProperty(LabelID As Long, VarName As String,
PropName As String) As String
```

Method:

```
Basic
```

See also:

LabelGetVarName, LabelGetVarCount, LabelSetVar

LabelOpen

Description:

Opens an existing label. As a parameter the label file name (LabelName) should be specified. If the label is successfully opened, a label ID is returned, which will later be used in other functions. If it doesn't succeed -1 if is returned. The reason for failure can be obtained from ErrorMessage and DetailedMessage functions.

PrinterName is optional parameter, and means that the label will be opened with that printer.

Syntax:

```
Function LabelOpen(FileName As String) As Long
```

Method:

```
Basic
```

See also:

LabelClose

LabelOpenEx

Description:

LabelOpenEx method returns the interface of label selected with the filename. For more information about INiceLabel please read INiceLabel chapter.

PrinterName is optional parameter, and means that the label will be opened with that printer.

Syntax:

```
Function LabelOpenEx(FileName As String) As IniceLabel
```

Method:

```
Advanced
```

LabelOpenExForPrinter

Description:

This method is the same as LabelOpenEx just that it opens the label with the specified printer (if the printer does not exist then the label is not opened)

Syntax:

```
Function LabelOpenExForPrinter(FileName As String, PrinterName As
String) As INiceLabel
```

Method:

```
Advanced
```

LabelOpenForPrinter

Description:
This method is the same as LabelOpen just that it opens the label with the specified printer (if the printer does not exist then the label is not opened)

Syntax:
```
Function LabelOpenForPrinter(FileName As String, PrinterName As
String) As Long
```

Method:

```
Basic
```

LabelPrint

Description:

This function prints the label with the specified label ID. Quantity must be represented as a string.

The quantity parameter is the print quantity of the labels; it can be a number, or one of following words:

- VARIABLE
- UNLIMITED

The first parameter means printing on the base of the variable quantity (one of the variables sets the quantity), the second one means unlimited printing (printing from the whole database file for example).

Skip, Identical and LabelSets are optional parameters. Skip parameter must be provided in order you want to skip a number of labels before printing. Identical parameter defines the amount of identical copies of the label. LabelSets is the number of label sets which you want to print.

Syntax:

```
Function LabelPrint(LabelID As Long, Quantity As String, Skip As
String, Identical As String, LabelSets As String) As Boolean
```

Example

```
Success = LabelPrint("1,1")
```

Method:

```
Basic
```

See also:

LabelSessionPrint, LabelSessionStart, LabelSessionEnd, LabelSetPrinter

LabelSessionEnd

Description:

The function ends print session.

Syntax:

```
Function LabelSessionEnd(LabelID As Long) As Boolean
```

See also:

LabelSessionStart, LabelSessionPrint, LabelPrint, LabelSetPrinter

Method:

Basic

LabelSessionPrint

Description:

You send the data stream to printer using this function.
This function prints the label with the specified label ID. Quantity must be represented as a string.

The quantity parameter is the print quantity of the labels; it can be a number, or one of following words:

- VARIABLE
- UNLIMITED

The first parameter means printing on the base of the variable quantity (one of the variables sets the quantity), the second one means unlimited printing (printing from the whole database file for example).

You can use multiple LabelSessionPrint commands one after another and join them in single data stream. The stream is not closed until the LabelSessionEnd occurs.

Syntax:

```
Function LabelSessionPrint(LabelID As Long, Quantity As String) As
Boolean
```

See also:

LabelSessionStart, LabelSessionEnd, Labelprint, LabelSetPrinter

Method:

Basic

LabelSessionStart

Description:

All three functions (LabelSessionStart, LabelSessionPrint, LabelSessionEnd) are used together. If ordinary command LabelSessionPrint is used, every time a complete data stream for printer is sent to printer. If you want to join multiple Print commands into one data stream, you can use the LabelSessionStart command followed with any number of LabelSessionPrint commands and in the end use the LabelSessionEnd command. The stream is not closed until the LabelSessionEnd occurs. These commands offer optimal printing ,because and it is not necessary to generate a complete data stream for each print session.
SessionStart method clean label variables. LabelSetVar has to be used after LabelSessionStart method.

Syntax:

```
Function LabelSessionStart(LabelID As Long) As Boolean
```

Method:

Basic

See also:

LabelSessionPrint, LabelSessionEnd, LabelPrint, LabelSetPrinter

LabelSetPrinter

Description:

This function changes the printer on which the label will be printed.

Syntax:

```
Function LabelSetPrinter(LabelID As Long, PrinterName As String)
As Boolean
```

Method:

```
Basic
```

See also:

LabelSessionPrint, LabelSessionEnd, LabelSessionStart, LabelPrint

LabelSetPrintJobName

Description:

Using this function you can set the Print job name that NiceLabel will use at the next print command. After printing the name is returned in normal state.

Syntax:

```
Function LabelSetPrintJobName(LabelID As Long, PrinterName As
String) As Boolean
```

Method:

```
Basic
```

LabelSetVar

Description:

The function sets the value of the specified variable. Label ID, variable name and its desired value must be specified. If you want, you can also set the Step and Count options for the automatic incrementing of variable. If you don't want to change that, specify -9999 for Count and Step.

Quantity:

```
Function LabelSetVar(LabelID As Long, Name As String, Value As
String, Step As Long, Count As Long) As Boolean
```

Method:

`Basic`

See also:

LabelGetvarCount, LabelGetVarName, LabelGetVarProperty

LabelTestConnection

Description:

With this function you can check the current link between LabelID and name of the loaded label file. In case the OLE connection was broken or NiceLabel application was closed, you can use this function as a tester for OLE server activity. If function returns false, this means that somebody has closed the label file and current LabelID should be discarded. You should also call LabelOpen function again.

Syntax:

```
Function LabelTestConnection(LabelID As Long, FileName As String)
As Boolean
```

Method:

```
Basic
```

Login

Description:

Performs login into NiceLabel if it has not been executed already.

Syntax:

```
Function Login(UserName As String, Level As Long) As Boolean
```

Method:

```
Basic, Advanced
```

ProcessID

Description:
Returns the process id of the application (in case user needs to "kill" it)

Syntax:
```
Property ProcessID As Long,  read-only
```

Access Rights:

read-only

Method:

```
Basic, Advanced
```

Quit

Description:

This procedure closes the NiceLabel if it was opened with the client application.

Syntax:

```
Quit
```

Method:

Basic, Advanced

SetDefaultPrinter

Description:

This method set the default printer for NiceLabel.

Returned result :
1 - the printer could not be set
0 - the printer could be set

Syntax:

Function SetDefaultPrinter(DefaultPrinterName As String) As Long

Method:

Basic, Advanced

SetApplicationProperty

Description:

Function enables dialogs to be shown when application is run under system service.

Syntax:

Function SetApplicationProperty(PropertyName As String, PropertyValue As String) As Long

Explanation of the parameters:

`PropertyName`

```
Name of the property that you want to set, only one property is available
at this time: 'ShowDialogsDuringAutomation'
```

`PropertyValue`

```
Value of the property, for the property 'ShowDialogsDuringAutomation'
value '1' means that the dialogs will be shown when application is run
through ActiveX, any other value means that the dialogs will not be shown
```

2.9.2 Hierarchy diagram

The diagram below shows the connection between all interfaces included in nice4.olb:

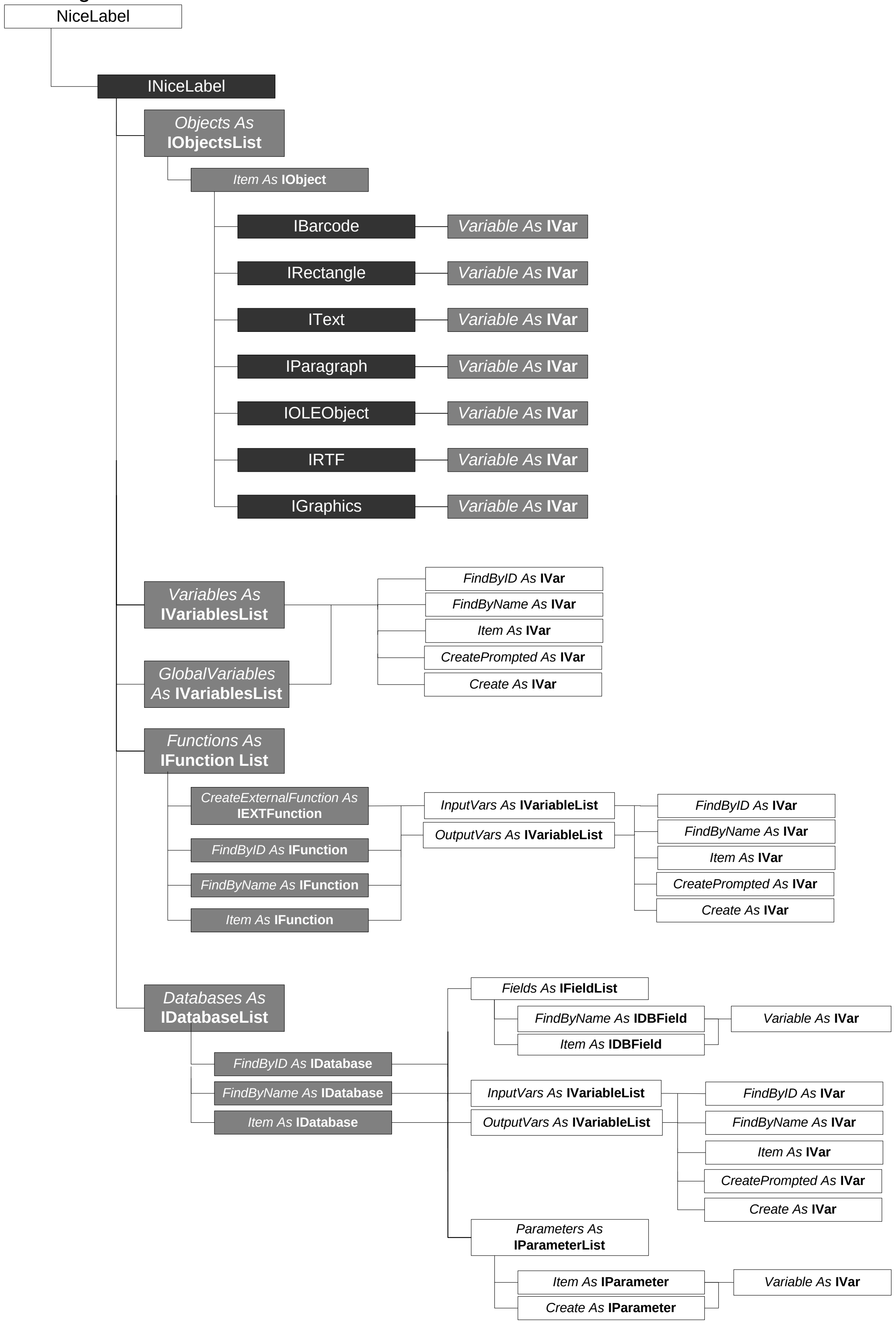


Figure 2: ActiveX interface hierarchy

2.9.3 Class IBarcode (Advanced only)

Methods	Version Availability	Properties	Version Availability
GetProperty	3,4,5	AnchorElementID	3,4
Move	3,4,5	AnchorLevel	3,4,5
Resize	3,4,5	AnchorPoint	3,4,5
SetContents	3,4,5	AutoCDCalculation	3,4,5
SetVariable	3,4,5	BarcodeType	3,4,5
		Color	4,5
		Contents	3,4,5
		ContentsMask	3,4,5
		FormatID	3,4,5
		HasQuietZone	3,4,5
		Height	3,4,5
		ID	3,4,5
		IncludeCD	3,4,5
		IsLocked	3,4,5
		Kind	3,4,5
		Left	3,4,5
		MaxLength	3,4,5
		Name	3,4,5
		PageNumber	3,4,5
		PrintAsGraphics	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		Selected	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values are:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

```
Hight, Left, Top, Width
```

AutoCDCalculation

Description:

When this property is true, CheckDigit for the barcode is calculated automatically – only the contents should be provided. When the property value is false, NiceLabel will verify the contents of the barcode, as the checkdigit is provided together with the data.

Syntax:

```
AutoCDCalculation As Boolean
```

See also:

`IncludeCD`

BarcodeType

Description:

Returns the Barcode Type.

Possible values are:

Value	Description
1	EAN13
2	EAN8
3	UPC_A
4	UPC_E
5	I2OF5
6	CODE39
7	CODE128A
8	CODE128B
9	CODE128C
10	CODABAR
11	CODE128
12	GS1-128
13	POSTNET32
14	POSTNET37
15	POSTNET52
16	POSTNET62
17	BOOKLAND
18	DISTRIBUTION
19	CODE93

20	NG_EAN13
21	NG_EAN8
22	PDF417
23	BC412
24	DATAMATRIX
25	MAXICODE
26	EAN128
27	AZTEC
28	QR
29	SSCC
30	MSI
31	CBLOCK_F
32	PHARMACODE
33	EXTCODE39
34	EAN14
35	KIX
36	ITF14
37	2DPHARMACODE
38	CODE32
39	EAN13+2
40	EAN13+5
41	EAN8+2
42	EAN8+5
43	UPC_A+2
44	UPC_A+5
45	UPC_E+2
46	UPC_E+5
47	ITF16
48	MICROPDF
49	GS1 Databar
50	Addon 2
51	Addon 5
52	Code 39 Tri Optik
53	Plessy
55	Anker
56	MicroQR

60	UPC Case Code
61	Dun 14
62	InfoGlyph
63	UPC E (tip 1)
66	SSCC18

Syntax:

```
Property BarcodeType As String
```

Access Rights:

read-only

See also:

Color, ContentsMask

<u>Color</u>

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
Color As Long
```

See also:

BarcodeType, ContentsMask

<u>Contents</u>

Description:

Returns the current contents of the element.

Syntax:

```
Property Contents As String
```

Access Rights:

read-only

ContentsMask

Description:

Sets and gets the ContentsMask attribute.

Syntax:

```
ContentsMask As String
```

See also:

BarcodeType, Color

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

GetPropery

Description:

Returns the property of a barcode.

Possible names are:

IsStructured – used for MaxiCode barcode

Syntax:

```
Function GetProperty(Name As String) As String
```

<u>HasQuietZone</u>

Description:

If this property is true, the size of the barcode object is enlarged by the quiet zone.

Syntax:

```
HasQuietZone As Boolean
```

<u>Height</u>

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Left, Top, Width`

<u>ID</u>

Description:

ID of the element

Syntax:

```
Property ID As Long
```

<u>IncludeCD</u>

Description:

For some barcode symbologies, CheckDigit is optional (Code39, I2of5). This property specifies, if checkdigit is to be included in the barcode or not.

Syntax:

```
IncludeCD As Boolean
```

See also:

AutoCDCalculation

IsLocked

Description:

When the element's position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

Kind

Description:

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

```
Hight, AnchorPoint, Top, Width
```

MaxLenght

Description:

Returns the maximum data length available for this barcode.

Syntax:

```
MaxLenght As Long
```

See also:

Name

Name

Description:

Name property represents the name of the barcode.

Syntax:

```
Name As String
```

See also:

MaxLenght

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

PageNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

See also:

PrintAsGraphics

PrintAsGraphics

Description:

When this property is set to TRUE, the barcode will always be printed as graphics, even if the printer supports printing it with an internal command.

Syntax:

```
PrintAsGraphics As Boolean
```

See also:

PageNumber

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

`ResizeFlag`

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

`Resize`

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

`Rotation`

Rotation

Description:

Specifies the element's rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
Rotation
```

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetContents

Description:

When the contents of an element should be changed, SetContents method should be called. In case of success (the Value is valid for the element), the function returns 0. In case of an error, the function returns –1.

Syntax:

```
Function SetContents(Value As String) As Long
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

See also:

Variable

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

`Hight, Left, AnchorPoint, Width`

Variable

Description:

Returns the interface to the variable, which is attached to the element.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

See also:

SetVariable

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

`Hight, Left, Top, AnchorPoint`

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

Access Rights:

read-only

2.9.4 Class IDatabase (Advanced Only)

Methods	Version Availability	Properties	Version Availability
SelectRecords	4,5	Alias	3,4,5
		DBPassword	3,4,5
		Delimiter	3,4,5
		Description	3,4,5
		DriverType	3,4,5
		EPDSConnection	3,4,5
		Fields	3,4,5
		ID	3,4,5
		InputVars	3,4,5
		IsDelimited	3,4,5
		IsUsed	4,5
		Kind	3,4,5
		Name	3,4,5
		Navigation	3,4,5
		Order	3,4,5
		OutputVars	3,4,5
		Parameters	3,4,5
		ReviewBeforePrint	3,4,5
		SelectedRecordCount	4,5
		Separator	3,4,5
		SQL	3,4,5
		Table	3,4,5

Alias

Description:

Alias represents the name of the alias for the database. See BDE Administrator.

Syntax:

```
Alias As String
```

DBPassword

Description:

DBPassword property represents the password for the database.

Syntax:

```
DBPassword As String
```

Delimiter

Description:

With this property you can select the ***Delimiter*** character that is used when separator character is used in the text field itself. The delimiter should be used to enclose such field. Text between two delimiter characters is treated as one field although it contains the field separator character.

Syntax:

```
Delimiter As String
```

Description:

With this property you can select the ***Separator*** character that is used for a border between two fields in a text file.

Syntax:

```
Separator As String
```

See also:

Separator, IsDelimited

DriverType

Description:

DriverType is in main driver name. You can find suitable names in BDE Administrator and also in NiceLabel Pro in Database dialog box under the driver selection.

Syntax:

```
DriverType As String
```

See also:

Name, Table, Order, ReviewBeforePrint

EPDSConnection

Description:

This property is defining the database connection, as is internally used by NiceLabel and other Nice programs.

Syntax:

```
Property EPDSConnection As String
```

Fields

Description:

Returns the interface to the fields which are present in the database.

Syntax:

```
Property Fields As IFieldList
```

Access Rights:

```
read-only
```

ID

Description:

ID of the database.

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

InputVars

Description:

Returns the interface to the variable list for input variables.

Syntax:

```
Property InputVars As IVariableList
```

Access Rights:

read-only

See also:

OutputVars

IsDelimited

Description:

If data fields in your text file are separated with some special character, you should set this property to TRUE value. In case that your data fields always occupy the same number of characters this property should be set to FALSE.

Syntax:

```
IsDelimited As Boolean
```

See also:

Separator, Delimiter

IsUsed

Description:
If this property is true then this function is used on the label

Syntax:

```
IsUsed As Boolean
```

Kind

Description:

Kind property for the database is always set to 4.

Value	Description
1	Concatenate
2	Subset
3	Linear Function
4	Database
5	EAN 128
6	CD Algo
7	Date Addition
8	Euro

9	External
10	SSCC

Syntax:

```
Property Kind As Long
```

Access Rights:

```
read-only
```

<u>Name</u>

Description:

In the Name property name of the database is stored. This property can be set in NiceLabel Pro on General tab of Database dialog box.

Syntax:

```
Name As String
```

See also:

DriverType, Table, Order, ReviewBeforePrint

<u>Navigation</u>

Description:

Navigation property represents how record retrieving will be set for selected database access.

Possible values for the Navigation property are:

Value	Description
0	First
1	All
2	Select
3	Last

Syntax:

```
Navigation As Long
```

See also:

SelectRecords, SelectRecordsCount

Order

Description:

Order property represents the name of the field which is used for sorting records.

Syntax:

```
Order As String
```

See also:

Name, Table, DriverType, ReviewBeforePrint

OutputVars

Description:

Returns the interface to the variable list for output variables.

Syntax:

```
Property OutputVars As IVariableList
```

Access Rights:

read-only

See also:

InputVars

Parameters

Description:

Returns the interface to the parameters (filters) which are present in the database.

Syntax:

```
Property Parameters As IParameterList
```

Access Rights:

```
read-only
```

See also:

SQL

ReviewBeforePrint

Description:

ReviewBeforePrint property's value is TRUE, if ReviewBeforePrint option is enabled in Database Access definition. This option makes it possible to change the result of a function just before using its results on the label.

Syntax:

```
ReviewBeforePrint As Boolean
```

See also:

Name, Table, Order, DriverType

SelectRecords

Description:

Execute this method if you would like to show NiceLabel's Record Selection dialog box, where the user has possibility to select which records from the database will be printed.

Parent parameter, which must be passed to the SelectRecords function is a handle to the parent window on which the Record Selection dialog box will appear.

Syntax:

```
Function SelectRecords(Parent As Long) As Long
```

Access Rights:

```
read-only
```

See also:

Navigation, SelectRecordsCount

SelectedRecordCount

Description:

Returns the number of selected records for the database, which is associated with this IDatabase interface. If no records are selected or database connection does not exists the property contains - 1

Syntax:

```
Property SelectedRecordCount As Long
```

Access Rights:

```
read-only
```

See also:

SelectRecords, Navigation

<u>Separator</u>

Description:

With this property you can select the ***Separator*** character that is used for a border between two fields in a text file.

Syntax:

```
Separator As String
```

See also:

IsDelimited, Delimiter

<u>SQL</u>

Description:

With SQL property you can select SQL statements for the database. Changing SQL statements can lead to failure in reading data from table.

Syntax:

```
SQL As String
```

Access Rights:

read-only

See also:

Parameters

<u>Table</u>

Description:

Table property represents the name of the table, which is used on this database access.

Syntax:

```
Table As String
```

See also:

Name, DriverType, Order, ReviewBeforePrint

2.9.5 Class IDatabaseList (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Item	3,4,5	Count	3,4,5
FindByID	3,4,5		
FindByName	3,4,5		

Count

Description:

Count property returnts the number of databases connected to the label.

Syntax:

```
Property Count As Long
```

Access Rights:

```
read-only
```

Item

Description:

Returns the interface to the database. Database is selected with Index.

Syntax:

```
Function Item(Index As Long) As IDatabase
```

FindByName

Description:

Returns the interface to the database. Database is selected with the name of the database.

Syntax:

```
Function FindByName(Name As String) As IDatabase
```

See also:

FindByID

FindByID

Description:

Returns the interface to the database. Database is selected with database ID.

Syntax:

```
Function FindByID(ID As Long) As IDatabase
```

See also:

FindByname

2.9.6 Class IDBDef

Methods	Version Availability	Properties	Version Availability
		Alias	3,4,5
		DBPassword	3,4,5
		Delimiter	3,4,5
		DriverType	3,4,5
		IsDelimited	3,4,5
		Separator	3,4,5

Alias

Description:

Alias represents the name of the alias for the database. See BDE Administrator.

Syntax:

```
Alias As String
```

DBPassword

Description:

DBPassword property represents the password for the database.

Syntax:

```
DBPassword As String
```

Delimiter

Description:

Property for setting or retrieving the delimiter char that is used in the text database.

Syntax:

```
Delimiter As String
```

DriverType

Description:

DriverType is in main driver name. You can find suitable names in BDE Administrator and also in NiceLabel Pro in Database dialog box under the driver selection.

Syntax:

```
DriverType As String
```

IsDelimited

Description:

If data fields in your text file are separated with some special character, you should set this property to TRUE value. In case that your data fields always occupy the same number of characters this property should be set to FALSE.

Syntax:

```
IsDelimited As Boolean
```

See also:

Separator

Separator

Description:

With this property you can select the ***Separator*** character that is used for a border between two fields in a text file.

Syntax:

```
Separator As String
```

See also:

`IsDelimited`

2.9.7 Class IDBField

Methods	Version Availability	Properties	Version Availability
<none>		IsSelected	3,4,5
		Length	3,4,5
		Name	3,4,5

		Number	3,4,5
		Offset	3,4,5
		Variable	3,4,5

IsSelected

Description:

When the field is selected for usage on the label, this property is TRUE.

Syntax:

```
Property IsSelected As Boolean
```

Access Rights:

read-only

Length

Description:

Length property represents the length of the field.

Syntax:

```
Property Length As Long
```

Access Rights:

read-only

Name

Description:

Name property represents the name of the field.

Syntax:

```
Property Name As String
```

Access Rights:

read-only

Number

Description:

Number property represents the number of the field.

Syntax:

```
Property Number As Long
```

Access Rights:

read-only

Offset

Description:

Offset property represents the offset of the beggining of the line. This property is available only for text files.

Syntax:

```
Property Offset As Long
```

Access Rights:

read-only

Variable

Description:

Returns the interface to the variable, which is connected with the database field.

Syntax:

```
Property Variable As IVar
```

Access Rights:

```
read-only
```

2.9.8 Class IDBParameter (Advanced Only)

Methods	Version Availability	Properties	Version Availability
SetVariable	3,4,5	Field	3,4,5
		Relation	3,4,5
		Variable	3,4,5

Field

Description:

Field property represents the name of the field which is used for filter.

Syntax:

```
Field As String
```

Relation

Description:

Relation property represents relation between field and a variable.

Possible values for Relation are:

Value	Description
0	==
1	>
2	>=
3	<
4	<=
5	<>
6	LIKE

Syntax:

```
Relation As Long
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

See also:

Variable

Variable

Description:

Returns the interface to the variable, which is connected with the database field.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

See also:

SetVariable

2.9.9 Class IEXTFunction (Advanced Only)

Methods	Version Availability	Properties	Version Availability
		Definition	3,4,5
		Description	3,4,5
		ID	3,4,5
		InputVars	3,4,5
		IsContentsProvider	3,4,5
		IsUsed	4,5
		Kind	3,4,5
		Name	3,4,5
		OutputVars	3,4,5
		ProviderName	4,5
		ReviewBeforePrint	3,4,5

Definition

Description:

Definition property represents a string, which consists a definiton for the function. Because of proper initialization of the label, this string is saved in a label file.

Syntax:

```
Definition As String
```

Description

Description:

In the Description property description of the function is stored.

Syntax:

```
Description As String
```

See also:

Name, ReviewBeforePrint

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

InputVars

Description:

Returns the interface to the variable list for input variables.

Syntax:

```
Property InputVars As IVariableList
```

Access Rights:

read-only

IsContentsProvider

Description:

If the function is contents provider for the element, then ISContentsProvider property has value TRUE.

Syntax:

```
Property IsContentsProvider As Boolean
```

Access Rights:

read-only

ProviderName

IsUsed

Description:

If this property is true then this function is used on the label

Syntax:

```
Property IsUsed As Boolean
```

Access Rights:

read-only

ProviderName

Kind

Description:

This property represents the value for the type of the function.

Possible values are:

Value	Description
1	Concatenate
2	Subset
3	Linear Function
4	Database
5	EAN 128
6	CD Algo
7	Date Addition
8	Euro
9	External
10	SSCC

Syntax:

```
Property Kind As Long
```

Acces Rights:

read-only

Name

Description:

Name property represents the name of the function.

Syntax:

```
Name As String
```

See also:

Description, ReviewBeforePrint

OutputVars

Description:

Returns the interface to the variable list for output variables.

Syntax:

```
Property OutputVars As IVariableList
```

Access Rights:

read-only

ProviderName

Description:

ProviderName represents the name of the component, which manage the execution and definition of the content's provider.

Syntax:

```
Property ProviderName As String
```

Access Rights:

read-only

See also:

IsContentsProvider

ReviewBeforePrint

Description:

ReviewBeforePrint property's value is TRUE, if ReviewBeforePrint option is enabled in function definition. This option makes it possible to change the result of a function just before using its results on the label.

Syntax:

```
ReviewBeforePrint As Boolean
```

Access Rights:

read-only

See also:

Name, Description

2.9.10 Class IFieldList (Advanced Only)

Methods	Version Availability	Properties	Version Availability
FindByName	3,4,5	Count	3,4,5
Item	3,4,5		

Count

Description:

Count property returns the number of fields, which are defined in the database.

Syntax:

```
Property Count As Long
```

Access Rights:

```
read-only
```

FindByName

Description:

Returns the interface to the field. Field is selected with the name of the field.

Syntax:

```
Function FindByName(Name As String) As IDBField
```

Item

Description:

Returns the interface to the field of the database. Field is selected with Index.

Syntax:

```
Function Item(Index As Long) As IDBField
```

2.9.11 Class IFunction (Advanced Only)

Methods	Version Availability	Properties	Version Availability
	4,5	Description	3,4,5
		ID	3,4,5
		InputVars	3,4,5

		Kind	3,4
		Name	3,4
		OutputVars	3,4
		ReviewBeforePrint	3,4

Description

Description:

In the Description property description of the function is stored.

Syntax:

```
Description As String
```

See also:

Name, ReviewBeforePrint

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

InputVars

Description:

Returns the interface to the variable list for input variables.

Syntax:

```
Property InputVars As IVariableList
```

Access Rights:

read-only

Kind

Description:

This property represents the value for the type of the function.

Possible values are:

Value	Description
1	Concatenate
2	Subset
3	Linear Function
4	Database
5	EAN 128
6	CD Algo
7	Date Addition
8	Euro
9	External
10	SSCC

Syntax:

```
Property Kind As Long
```

Acces Rights:

read-only

Name

Description:

Name property represents the name of the function.

Syntax:

```
Name As String
```

See also:

Description, ReviewBeforePrint

OutputVars

Description:

Returns the interface to the variable list for output variables.

Syntax:

```
Property OutputVars As IVariableList
```

Access Rights:

read-only

ReviewBeforePrint

Description:

ReviewBeforePrint property's value is TRUE, if ReviewBeforePrint option is enabled in function definition. This option makes it possible to change the result of a function just before using its results on the label.

Syntax:

```
ReviewBeforePrint As Boolean
```

See also:

Name, Description

2.9.12 Class IFunctionList (Advanced Only)

Methods	Version Availability	Properties	Version Availability
CreateExternalFunction	3,4,5	Count	3,4,5
FindByID	3,4,5		
FindByName	3,4,5		
Item	3,4,5		

Count

Description:

Count property returns the number of functions, which are defined on the label.

Syntax:

```
Property Count As Long
```

Access Rights:

```
read-only
```

CreateExternalFunction

Description:

This method creates an external function. Output is an interface to the new function.

Syntax:

```
Function CreateExternalFunction() As IExtFunction
```

<u>FindByID</u>

Description:

Returns the interface to the function. Function is selected with the ID of the function.

Syntax:

```
Function FindByID(ID As Long) As IFunction
```

See also:

FindByName

<u>FindByName</u>

Description:

Returns the interface to the function. Function is selected with the name of the function.

Syntax:

```
Function FindByName(Name As String) As IFunction
```

See also:

FindByID

<u>Item</u>

Description:

Returns the interface to the function. Function is selected with Index.

Syntax:

```
Function Item(Index As Long) As IFunction
```

2.9.13Class IGraphics (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Load	3,4,5	AnchorElementID	3,4,5
Move	3,4,5	AnchorLevel	3,4,5
Resize	3,4,5	AnchorPoint	3,4,5
SetVariable	3,4,5	FileName	3,4,5
		FormatID	3,4,5

		Height	3,4,5
		HorzMirror	3,4,5
		ID	3,4,5
		ImageType	4,5
		IsLocked	3,4,5
		Kind	3,4,5
		Left	3,4,5
		Name	3,4,5
		OnMemoryCard	3,4,5
		PageNumber	3,4,5
		ResizeFlag	3,4,5
		ResizeMode	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		Selected	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4,5
		VertMirror	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

Hight, Left, Top, Width

FileName

Description:

FileName property represents the file name of the graphics file.

Syntax:

```
Property FileName As String
```

Access Rights:

read-only

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

Left, AnchorPoint, Top, Width

HorzMirror

Description:

In case that HorzMirror property has value TRUE picture is mirrored horizontally. In the other case picture remains unmirrored.

Syntax:

```
HorzMirror As Boolean
```

See also:

VertMirror

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

ImageType

Description:

The method returns the type of the graphics
(0 , GF_UNKNOWN
 1, GF_DIB - device independant bitmap
 2, GF_METAFILE - metafile
 3, GF_IMK - memory card picture
 4, GF_E_METAFILE - enhanced metafile
)

Syntax:

```
Property ImageType As Long
```

Access Rights:

read-only

IsLocked

Description:

When the element's position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

Kind

Description:

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

Hight, AnchorPoint, Top, Width

Load

Description:

With Load method you can load the graphics. Filename is used for setting the file name of the graphics file.

Syntax:

```
Function Load(FileName As String) As Long
```

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

Name

Description:

Name property represents the name of the graphics.

Syntax:

```
Name As String
```

OnMemoryCard

Description:

In case that graphics is stored on a memory card then OnMemoryCard propery has value TRUE.

Syntax:

```
OnMemoryCard As Boolean
```

PageNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

<u>ResizeMode</u>

Description:

With ResizeMode it is possible to set the way of resizing the graphics.

Possible values for ResizeMode are:

Value	Description
0	Original si1ze
1	Proportional resize
2	Fit to size.

Syntax:

```
ResizeMode As Long
```

<u>RotateFlag</u>

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation

0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only¸

See also:

Rotation

<u>Rotation</u>

Description:

Specifies the element's rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

RotateFlag

<u>Selected</u>

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

See also:

Variable

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

Hight, AnchorPoint, Left, Width

Variable

Description:

Returns the interface to the variable, which is connected with the database field.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

See also:

SetVariable

VertMirror

Description:

In case that VertMirror property has value TRUE picture is mirrored vertically. In the other case picture remains unmirrored.

Syntax:

```
VertMirror As Boolean
```

See also:

HorzMirror

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

Hight, AnchorPoint, Top, Left

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

2.9.14Class ILabel (Advanced Only)

Methods	Version Availability	Properties	Version Availability
DatabaseProperties	3,4,5	AspectRatio	3,4,5
DeleteDatabase	3,4,5	CurrUnit	3,4,5
DeleteFunction	3,4,5	Databases	3,4,5
DeleteObject	3,4,5	Functions	3,4,5
DeleteVariable	3,4,5	GlobalVariables	3,4,5
DownloadVariablesList	5	IsLabelReadOnly	3,4,5
DownloadToPrinter	5	IsLinkedToStock	3,4,5
ExportToPrinter	4,5	IsStoreRecallPrintingAvailable	5
FunctionProperties	3,4,5	IsStoreRecallSupported	5
GetDownloadLocationsList	5	LabelSettings	3,4,5
GetDownloadLocationsListForPrinter	5	LastRevisionID	3,4,5
GeneralExport	4,5	Objects	3,4,5
GetCurrentState	4,5	PrinterName	3,4,5
GetDependantFilesList	3,4,5	PrinterPort	3,4,5
GetLabelPreviewEx	3,4,5	PrintType	5
GetPrinterDarknessList	3,4,5	ResX	3,4,5
GetPrintParamEx	5	ResY	3,4,5
GetPrintParamList	5	StockName	3,4,5
GetPrinterSpeedList	3,4,5	StockType	5
GetPrinterSpeedUnit	3,4,5	SynchronizedPrinting	3,4,5
GetPrintParam	3,4,5	Variables	3,4,5

NewDatabase	3,4,5		
NewDatabaseWiz	3,4,5		
NewFunction	3,4,5		
NewVariable	3,4,5		
NewVariableWiz	3,4,5		
ObjectProperties	3,4,5		
PrintAndGetJobID	5		
SelectDatabase	3,4,5		
SelectVariable	3,4,5		
SetDirty	4,5		
SetPrintJobName	4,5		
SetPrintParam	3,4,5		
VariableProperties	3,4,5		

AspectRatio

Description:

AspectRatio is a Boolean property which returns TRUE if aspect ratio is used otherwise FALSE is returned.

Syntax:

```
AspectRatio As Boolean
```

CurrUnit

Description:

CurrUnit represent the current measurement unit used in the label.

Possible values for are:

Value	Description
0	Cm
1	Inch
2	Dot
3	Mm

Syntax:

```
Property CurrUnit As Long
```

Access Rights:

read-only

DatabaseProperties

Description:

DatabaseProperties method opens a dialog box with properties for the database selected with ID.

Syntax:

```
Function DatabaseProperties(ID As Long) As Boolean
```

See also:

Databases, Deletedatabase, NewDatabase, NewDatabaseWiz, SelectDatabase

Databases

Description:

Returns the interface to the database list.

Syntax:

```
Property Databases As IDatabaseList
```

Access Rights:

read-only

See also:

DatabaseProperties, Deletedatabase, NewDatabase, NewDatabaseWiz, SelectDatabase

DeleteDatabase

Description:

DeleteDatabase method deletes a database connection from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteDatabase(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

DatabaseProperties, DataBases, NewDatabase, NewDatabaseWiz, SelectDatabase

DeleteFunction

Description:

DeleteFunction method deletes a function from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteFunction(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

Functions, FunctionProperties

DeleteObject

Description:

DeleteObject method deletes an object from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteObject(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

Objects, ObjectProperties

DeleteVariable

Description:

DeleteVariable method deletes a variable from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteVariable(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

DownloadVariablesList

Description:

This method is set a list of variables whose value must be set if we wish to call any export or Store to printer. If this is not done Export or Store to printer will fail.

Syntax:

```
Function DownloadVariablesList() As IVariableList
```

Avaliabilty:

Nicelabel Pro only

DownloadToPrinter

Description:

This method store label to printer. It is essential that we set all the variables in the list using DownloadVariablesList method or method will fail.

Syntax:

```
Function DownloadToPrinter() As Boolean
```

Avaliabilty:

Nicelabel Pro only

ExportToPrinter

Description:

Use ExportToPrinter function when you want export the label directly to the printer or to a file. When ExportFile parameter is blank exporting is done directly to the printer otherwise it is done to the file.

With CreateLVR parameter you are defining whether you want export also the LVR file, which contains the descriptions of the variables.

The ExportVariant defines how it is exported. This parameter depends on the printer (Default = 1, other values represent for example: export to RAM, export to Flash, export to PCMCIA, ...)

Syntax:

```
Function ExportToPrinter(ExportVariant As Long, ExportFile As
String, CreateLVR As Boolean) As Boolean
```

Functions

Description:

Returns the interface to the function list.

Syntax:

```
Property Functions As IFunctionList
```

Access Rights:

read-only

See also:

DeleteFunctions, FunctionProperties

FunctionProperties

Description:

FunctionProperties method opens a dialog box with properties for the function selected with ID.

Syntax:

```
Function FunctionProperties(ID As Long) As Boolean
```

See also:

DeleteFunctions, Functions

GetDownloadLocationsList

Description:
Returns available slot names for recall printing. Current printer name of the selected label is used.

Syntax:
```
Function GetDownloadLocationsList() As String
```

GetDownloadLocationsListForPrinter

Description:
Returns available slot names for recall printing. PrinterName is used as printer name

Syntax:
```
Function GetDownloadLocationsListForPrinter(PrinterName As String)
As String
```

General Export

Description:
Performs export of the current label to the ExportDir folder

Syntax:
Function GeneralExport(ExportDir As String) As Boolean

GetCurrentState

Description:

Use GetCurrentState function when you want to get the state of NiceLabel. Return result can be of the following:

1 - printing

2 - print preview

0 - other (label design)

This method is especially useful for VBScript actions, where the code can check current state and do some actions only in certain modes (printing for example)

Syntax:

```
Function GetCurrentState() As Long
```

GetDependantFilesList

Description:

GetDependantFilesList function returns you the list of files, which are in use with opened file.

As ListType you can set , which types of the file you wish to get. Below you can find all possible ListType parameters:

ListType = 0	All files
ListType = 1	NiceForm files
ListType = 2	Graphics files only
ListType = 3	Database files only

Returned string is a comma delimited string. This means that in one string there are names of all files in use. Return result is like this:

“C:\program files\EuroPlus\formsample1.xff, C:\program files\EuroPlus\formsample2.xff”

Syntax:

```
Function GetDependantFilesList(ListType As Long) As String
```

GetLabelPreviewEx

Description:

Use this function to generate the preview of the label. With this function it is possible to create label previews for the header and tail labels also. Also two sided labels are supported.

The extension of the FileName parameter defines in which format the preview will be stored. Possible formats are `BMP, JPG, PNG and GIF.`

Based on the parameters suitable preview is generated.

Syntax:

```
Function GetLabelPreviewEx(FileName As String, Width As Long,
Height As Long, LabelKind As Long, LabelSide As Long, HasBorder As
Boolean) As Boolean
```

Explanation of the parameters:

`FileName`

`Represents the file name.`

`Width`

`Width of picture in pixels.`

`Height`

`Height of picture in pixels.`

`LabelKind`

`With this parameter you specify which kind of label you want to preview. Possible values are 0, 1 and 2. 0 means header label, 1 means normal label and 1 means footer label.`

`LabelSide`

`With this parameter you specify which side of label you want to preview. Possible values are 0 and 1. 0 means front side of the label and 1 means the back side of the label.`

`HasBorder`

`If you specify this parameter the preview will have a border around the preview.`

GetPrinterDarknestList

Description:

This function returns the available darkness for the printer. The name of the printer must be provided as a parameter.

Syntax:

```
Function GetPrinterDarknessList(PrinterName As String) As String
```

See also:

GetPrinterSpeedList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GetPrintParamEX

Description:

This function retrieve the value of printing parameter.

Explanation of the parameters:

PrinterName
Defines for which printer you want to retrieve the parameter, blank value means that default printer will be taken.

ParamName

```
Defines which parameter should be retrieved :
PRINTSPEED – speed parameter as string value
PRINTDARKNESS - darkness parameter as string value
PRINTROTATION - rotation parameter as string value
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 - darkness on specific print head on the printer
                 with multiple print heads
NUMBER_OF_FRONT_PAGE_COLORS - number of colors for the specified
                              page side for the printers
                              with multiple print heads
NUMBER_OF_BACK_PAGE_COLORS - number of colors for the specified
                             page side for the printers
                             with multiple print heads
```

Syntax:

```
Function GetPrintParamEx(PrinterName As String, ParamName As
String) As String
```

See also:

GetPrinterSpeedList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GetPrintParamList

Description:

This method returns comma separated list of values/names for specified ParamName and specified PrinterName and the data you desire

Explanation of the parameters:

PrinterName
Defines for which printer you want to retrieve the parameter, blank value means that default printer will be taken.

ParamName

```
It can be one of :
PRINTSPEED – speed parameter as string value
PRINTDARKNESS - darkness parameter as string value
PRINTROTATION - rotation parameter as string value
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 - darkness on specific print head on the printer
                 with multiple print heads
FRONT_PAGE_COLOR_LIST - comma separated list of color values represented
                        in integers(COLORREF)
BACK_PAGE_COLOR_LIST - comma separated list of color values represented
                       in integers(COLORREF)
```

DataType
Parameter is used for parameters PRINTSPEED, PRINTDARKNESS, PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4, PRINTDARKNESS5 and it means whether you want to retrieve values list or names list for the specified parameter. Values are actual values for the speed/darkness while names are string representations for those items (for example you use names when displaying this data to user)

Syntax:

```
Function GetPrintParamList(PrinterName As String, ParamName As
String, DataType As Long) As String
```

See also:

GetPrinterSpeedList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GetPrinterSpeedList

Description:

This function returns the available speeds for the printer. The name of the printer must be provided as a parameter.

Syntax:

```
Function GetPrinterSpeedList(PrinterName As String) As String
```

See also:

GetPrinterDarknessList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GetPrinterSpeedUnit

Description:

This function returns the available speed units for the printer. The name of the printer must be provided as a parameter.

Syntax:

```
Function GetPrinterSpeedUnit(PrinterName As String) As String
```

See also:

GetPrinterDarknessList, GetPrinterSpeedList, GetPrintParam, PrinterName, PrinterPort

GetPrintParam

Description:

GetPrintParam function returns the value for provided printing parameter. Possible values for Name parameter are:

PRINTSPEED

Function will return the speed parameter as string value.

PRINTDARKNESS

Function will return the darkness parameter as string value.

Syntax:

```
Function GetPrintParam(Name As String) As String
```

See also:

GetPrinterDarknessList, GetPrinterSpeedList, Get PrinterSpeedUnit, PrinterName, PrinterPort

GlobalVariables

Description:

Returns the interface to the variable list for global variables.

Syntax:

```
Property GlobalVariables As IVariableList
```

Access Rights:

read-only

IsLabelReadOnly

Description:

IsLabelReadOnly property returns TRUE if label is opened in read only mode, otherwise return value is FALSE.

Syntax:

```
Property IsLabelReadOnly As Boolean
```

Access Rights:

`read-only`

IsLinkedToStock

Description:

IsLinkedToStock property returns TRUE if label is linked with some stock or FALSE if none of the stocks are used.

Syntax:

```
Property IsLinkedToStock As Boolean
```

Access Rights:

```
read-only
```

IsStoreRecallPrintingAvailable

Description:

If store recall printing is avaliable value of the IsStoreRecallPrintingAvaliable property is TRUE.

Syntax:

```
Property IsStoreRecallPrintingAvailable As Boolean
```

Access Rights:

```
read-only
```

IsStoreRecallSupported

Description:

If current printer supports store/recall value of the IsStoreRecallSupported property is TRUE.

Syntax:

```
Property IsStoreRecallSupported As Boolean
```

Access Rights:

```
read-only
```

LabelSettings

Description:

Returns an interface to IlabelSettings class. See more in the Ilabel Chapter.

Syntax:

```
Property LabelSettings As ILabelSettings
```

Access Rights:

```
read-only
```

LastRevisionID

Description:

Returns the ID of the last revision, which was made to the label.

Syntax:

```
Property LastRevisionID As Long
```

Access Rights:

```
read-only
```

NewDatabase

Description:

This method creates a new database access on the label.

Syntax:

```
Function NewDatabase() As Boolean
```

See also:

DatabaseProperties, DataBases, DeleteDatabases, NewDatabaseWiz, SelectDatabase

NewDatabaseWiz

Description:

NewDatabaseWiz method opens Database Wizard. In case of success method returns value TRUE.

Syntax:

```
Function NewDatabaseWiz() As Boolean
```

See also:

DatabaseProperties, DataBases, DeleteDatabases, NewDatabase, SelectDatabase

NewFunction

Description:

This method creates a new function on the label.

Syntax:

```
Function NewFunction() As Boolean
```

NewVariable

Description:

This method creates a new variable on the label.

Syntax:

```
Function NewVariable() As Boolean
```

See also:

`NewVariableWiz, Variables, VariableProperties`

NewVariableWiz

Description:

NewVariableWiz method opens Variable Wizard. In case of success method returns value TRUE.

Syntax:

```
Function NewVariableWiz() As Boolean
```

See also:

`NewVariable, Variables, VariableProperties`

Objects

Description:

Returns the interface to the object list.

Syntax:

```
Property Objects As IObjectList
```

Access Rights:

read-only

See also:

DeleteObjects, ObjectProperties

ObjectProperties

Description:

ObjectProperties method opens a dialog box with properties for the object selected with ID.

Syntax:

```
Function ObjectProperties(ID As Long) As Boolean
```

See also:

Objects, DeleteObject

PrintAndGetJobID

Description:
When printing from NiceLabel user can retrieve an exact status of the print job.
Parameter Quantity : Specifies the quantity that should be used when printing
AdditionalParam: Reserved for future use, must be “
Return Value : Job ID of the print Job.
In case of error, return value can contain the following strings:
“Logging Disabled” - Logging is not enabled, retrieving of ID not possible.
“Print Failed” - Print Job failed to print.

Syntax:

```
Function PrintAndGetJobID(Quantity As String, AdditionalParams As
String) As String
```

PrinterName

Description:

PrinterName represents the name of the printer to which label will be printed.

Syntax:

```
PrinterName As String
```

See also:

GetPrinterDarknessList, GetPrinterSpeedList, GetPrintParam, PrinterPort, GetPrintParam

PrinterPort

Description:

Command PrinterPort allows you to set different printer port than specified in the label file.

Syntax:

```
PrinterPort As String
```

Access Rights:

```
read-write
```

See also:

GetPrinterDarknessList, GetPrinterSpeedList, GetPrintParam, PrinterName, GetPrintParam

PrintType

Description:

Label mode can be obtained by calling PrintType property
O = PRINT_TYPE_NORMAL
1 = PRINT_TYPE_RECALL

Syntax:

```
PrintType As Long
```

Access Rights:

```
read
```

ResX

Description:

Returns the current printer resolution in horizontal direction. Result is in DPI.

Syntax:

```
Property ResX As Long
```

Access Rights:

read-only

See also:

```
ResY
```

ResY

Description:

Returns the current printer resolution in vertical direction. Result is in DPI.

Syntax:

```
Property ResY As Long
```

Access Rights:

read-only

See also:

```
ResX
```

SelectDatabase

Description:

SelectDatabase method selects a database on the label. In case of success method returns value TRUE.

Syntax:

```
Function SelectDatabase(ID As Long) As Boolean
```

See also:

DatabaseProperties, DataBases, DeleteDatabases, NewDatabase, NewDatabaseWiz

SelectVariable

Description:

SelectVariable method selects a variable on the label. In case of success method returns value TRUE.

Syntax:

```
Function SelectVariable(ID As Long) As Boolean
```

SetDirty

Description:

With this method you can set the dirty flag of the label document to true, if you do this means that the label has been modifed and the user will be prompted to save the label when he will try to close the label

Syntax:

```
Function SetDirty(flag As Boolean)
```

SetPrintJobName

Description:

The INiceLabel interface is extended to support setting the PrintJobName for next print command. The command is defined as:

Parameter JobName must contain the name of the print job. function returns true if successful.

The parameter set is used only for the first print command. After that it is reset to use default print job names. Application should set the print job name each time before executing print commands.

SetPrintParam

Description:

With SetPrintParam function it is possible to set the following printing parameters:

- SPEED – Name parameter : PRINTSPEED
- DARKNESS – Name parameter : PRINTDARKNESS
- PAPER BIN – Name parameter : PAPERBIN
- ROTATION – Name parameter : PRINTROTATION
  Value : 0 - no rotation
  Value : 1 - 180° rotation
- LEFT OFFSET FOR ALL PRINTING OBJECTS – Name Parameter: PRINTOFFSETX
  Value: Numeric positive or numeric negative in pixels
- TOP OFFSET FOR ALL PRINTING OBJECTS – Name Parameter: PRINTOFFSETY
  Value: Numeric positive or numeric negative in pixels

The name of the printing parameter should be stored in the function's Name parameter and the value of the parameter in the Value parameter. Both parameters should be represented as a string. If function is able to set the required printing parameters, the return value will be TRUE otherwise something was wrong and the return result will be FALSE.

Syntax:

```
Function SetPrintParam(Name As String, Value As String) As Boolean
```

StockName

Description:

Returns the name of the stock, which is used on the label.

Syntax:

```
Property StockName As String
```

Access Rights:

`read-only`

See also:

`StockType`

StockType

Description:

Returns the type of the stock, which is used on the label.

Syntax:

```
Property StockType As String
```

Access Rights:

`read-only`

See also:

`StockName`

SynchronizedPrinting

Description:

This property defines wether the synchronized printing should be used or not. If you try to set this property to true when the current printer does not support this then the property will be false and error will be set in NiceLabel.

Syntax:

```
Property SynchronizedPrinting As Boolean
```

Variables

Description:

Returns the interface to the variable list for input variables.

Syntax:

```
Property Variables As IVariableList
```

Access Rights:

read-only

See also:

`NewVariable, NewVariableWiz, VariableProperties`

UpdateVariableData

Description:

UpdateVariableData method change global variable setup.

Syntax:

VariableProperties

Description:

VariableProperties method opens a dialog box with properties for the variable selected with ID.

Syntax:

```
Function VariableProperties(ID As Long) As Boolean
```

See also:

`NewVariable, NewVariableWiz, Variables`

2.9.15 Class INiceLabel (Advanced Only)

Methods	Version Availability	Properties	Version Availability
CustomExport	5	Application	3,4,5
DatabaseProperties	3,4,5	CurrUnit	3,4,5
DeleteDatabase	3,4,5	Databases	3,4,5
DeleteFunction	3,4,5	Functions	3,4,5
DeleteObject	3,4,5	GlobalVariables	3,4,5
DeleteVariable	3,4,5	IsLabelReadOnly	3,4,5
DownloadVariablesList	5	IsLinkedToStock	3,4,5
DownloadToPrinter	5	IsStoreRecallPrintingAvailable	5
ExecuteMacro	3,4,5	IsStoreRecallSupported	5
ExportToPocket	3,4,5	LabelFileName	3,4,5
ExportToPrinter	3,4,5	LabelSettings	3,4,5
FunctionProperties	3,4,5	LastRevisionID	3,4,5
GeneralExportEX	5	Objects	3,4,5
GeneralExport	3,4,5	PrinterName	3,4,5

GetCurrentState	4,5	PrinterPort	3,4,5
GetDependantFilesList	3,4,5	PrintType	5
GetDownloadLocations List	5	SavedPrinterName	5
GetDownloadLocations ListForPrinter	5	StockName	3,4,5
GetLabelPreview	3,4,5	StockType	3,4,5
GetLabelPreviewEx	4,5	SynchronizedPrinting	5
GetPrintParam	3,4,5	Variables	3,4,5
GetPrintParamEx	5		
GetPrintParamList	5		
NewDatabase	3,4,5		
NewDatabaseWiz	3,4,5		
NewFunction	3,4,5		
NewVariable	3,4,5		
NewVariableWiz	3,4,5		
ObjectProperties	3,4,5		
Print	3,4,5		
PrintAndGetJobID	5		
Save	3,4,5		
SelectDatabase	3,4,5		
SelectVariable	3,4,5		
SessionEnd	3,4,5		
SessionPrint	3,4,5		
SessionStart	3,4,5		
SetDirty	4,5		
SetObjectVisible	5		
SetPrintJobName	4,5		
SetPrintParam	3,4,5		
VariableProperties	3,4,5		

Application

Description:

Application method returns an interface to the niceApp.

Syntax:

```
Property Application As NiceLabel
```

Access Rights:

`read-only`

<u>CustomExport</u>

Description:

This method enables label export using custom export definition. Definition has to be previous defined via Export Print system menu

Syntax:

```
Function CustomExport(ExportName As String, ExportDir As String)
As Boolean
```

Explanation of the parameters:

```
ExportName – custom export definition name
ExportDir – folder path of the exported file
```

<u>CurrUnit</u>

Description:

CurrUnit represent the current measurement unit used in the label.

Possible values are:

Value	Description
0	Cm
1	Inch
2	Dot
3	Mm

Syntax:

```
Property CurrUnit As Long
```

Access Rights:

read-only

Databases

Description:

Returns the interface to the database list.

Syntax:

```
Property Databases As IDatabaseList
```

Access Rights:

read-only

See also:

`DatabaseProperties, DeleteDatabase, NewDatabase, NewDatabaseWiz, SelectDatabase`

DatabaseProperties

Description:

DatabaseProperties method opens a dialog box with properties for the database selected with ID.

Syntax:

```
Function DatabaseProperties(ID As Long) As Boolean
```

See also:

`Databases, DeleteDatabase, NewDatabase, NewDatabaseWiz, SelectDatabase`

DeleteDatabase

Description:

DeleteDatabase method deletes a database connection from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteDatabase(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

`Databases, DatabaseProperties, NewDatabase, NewDatabaseWiz, SelectDatabase`

DeleteFunction

Description:

DeleteFunction method deletes a function from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteFunction(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

`Functions, FunctionProperties, NewFunction`

DeleteObject

Description:

DeleteObject method deletes an object from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteObject(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

`Objects, ObjectPropeties`

DeleteVariable

Description:

DeleteVariable method deletes a variable from the label. In case of success method returns value TRUE.

Syntax:

```
Function DeleteVariable(ID As Long) As Boolean
```

Avaliabilty:

Nicelabel Pro only

See also:

`GlobalVariables, NewVariable, SelectVariable, Variables, VariableProperties, NewVariableWiz`

DownloadVariablesList

Description:

This method is set a list of variables whose value must be set if we wish to call any export or Store to printer. If this is not done

Syntax:

```
Function DownloadVariablesList() As IVariableList
```

Avaliabilty:

Nicelabel Pro only

DownloadToPrinter

Description:

This method store label to printer. It is essential that we set all the variables in the list using DownloadVariablesList method or method will fail.

Syntax:

```
Function DownloadToPrinter() As Boolean
```

Avaliabilty:

Nicelabel Pro only

ExecuteMacro

Description:

This function executes a macro. Macro parameter represents NiceCommand.

Syntax:

```
Function ExecuteMacro(Macro As String) As Boolean
```

ExportToPocket

Description:

Function executes the same action as choosing the command **File|Export|Export to terminal**. All the necessary files are created in directory specified in ExportDir parameter.

Function is not supported in DEMO mode.

Syntax:

```
Function ExportToPocket(ExportDir As String) As Boolean
```

See also:

`ExportToPrinter`

ExportToPrinter

Description:

Use ExportToPrinter function when you want export the label directly to the printer or to a file. When ExportFile parameter is blank exporting is done directly to the printer otherwise it is done to the file.

With CreateLVR parameter you are defining whether you want export also the LVR file, which contains the descriptions of the variables.

The ExportVariant defines how it is exported. This parameter depends on the printer (Default = 1, other values represent for example: export to RAM, export to Flash, export to PCMCIA, ...)

Function is not supported in DEMO mode.

Syntax:

```
Function ExportToPrinter(ExportVariant As Long, ExportFile As
String, CreateLVR As Boolean) As Boolean
```

See also:

`ExportToPocket`

Functions

Description:

Returns the interface to the function list.

Syntax:

```
Property Functions As IFunctionList
```

Access Rights:

read-only

See also:

`DeleteFunction, FunctionProperties, NewFunction`

FunctionProperties

Description:

FunctionProperties method opens a dialog box with properties for the function selected with ID.

Syntax:

```
Function FunctionProperties(ID As Long) As Boolean
```

See also:

```
DeleteFunction, Functions, NewFunction
```

GetDependantFilesList

Description:

GetDependantFilesList function returns you the list of files, which are in use with opened file.

As ListType you can set , which types of the file you wish to get. Below you can find all possible ListType parameters:

ListType = 0	All files
ListType = 1	NiceForm files
ListType = 2	Graphics files only
ListType = 3	Database files only

Returned string is a comma delimited string. This means that in one string there are names of all files in use. Return result is like this:

“C:\program files\EuroPlus\formsample1.xff, C:\program files\EuroPlus\formsample2.xff”

Syntax:

```
Function GetDependantFilesList(FileName As String, Width as Long,
Height as Long) As Boolean
```

GetDownloadLocationsList

Description:

Returns available slot names for recall printing. Current printer name of the selected label is used.

Syntax:

```
Function GetDownloadLocationsList() As String
```

GetDownloadLocationsListForPrinter

Description:

Returns available slot names for recall printing. PrinterName is used as printer name

Syntax:

```
Function GetDownloadLocationsListForPrinter(PrinterName As String)
As String
```

GeneralExport

Description:

Performs export of the current label to the ExportDir folder.

Function is not supported in DEMO mode.

Syntax:

```
Function GetLabelPreview(ExportDir As String) As Boolean
```

GeneralExportEx

Description:

GeneralExportEx method is introduced primarily to support exporting of store/recall streams..

Syntax:

```
Function GeneralExportEx(ExportDir As String, Parameters As
String) As
```

Parameters

ExportDir
Export file will be created into this directory.

Parameters

```
Parameters for store/recall can be specified, if the parameters is empty
the method behaves like normal GeneralExport method.
Available options for Parameters:
- RecallPrinting=YES – you get recall stream
                 NO  - you get no recall stream
- StoreLocation=SampleStoreLocation - StoreLocation is a string
representing actual memory location in printer, this should be used
together with the RecallPrint option
- NormalPrinting=YES - when the label is set to recall printing and you
                       want to get normal print stream
```

```
                NO - when the label is set to recall printing and you
                     not want to get normal print stream
Options must be separated with CR LF characters.
```

Example for recall stream:

```
RecallPrinting=YES
StoreLocation=Define Form
```

Example for normal printing when recall mode is saved in the label:

```
NormalPrinting=YES
```

GetCurrentState

Description:

Use GetCurrentState function when you want to get the state of NiceLabel. Return result can be of the following:

1 - printing

2 - print preview

0 - other (label design)

This method is especially useful for VBScript actions, where the code can check current state and do some actions only in certain modes (printing for example)

Syntax:

```
Function GetCurrentState() As Long
```

GetLabelPreview

Description:

GetLabelPreview method saves a preview of the label in the file specified with the FileName. Other two parameters are for defining the size of preview. If file is created sucessfully return value is TRUE.

The extension of the FileName parameter defines in which format the preview will be stored. Possible formats are `BMP, JPG, PNG and GIF.`

Syntax:

```
Function GetLabelPreview(FileName As String, Width As Long, Height
As Long) As Boolean
```

GetLabelPreviewEx

Description:

Use this function to generate the preview of the label. With this function it is possible to create label previews for the header and tail labels also. Also two sided labels are supported.

The extension of the FileName parameter defines in which format the preview will be stored. Possible formats are `BMP, JPG, PNG and GIF.`

Based on the parameters suitable preview is generated.

Syntax:

```
Function GetLabelPreviewEx(FileName As String, Width As Long,
Height As Long, LabelKind As Long, LabelSide As Long, HasBorder As
Boolean) As Boolean
```

Explanation of the parameters:

`FileName`

`Represents the file name.`

`Width`

`Width of picture in pixels.`

`Height`

`Height of picture in pixels.`

`LabelKind`

`With this parameter you specify which kind of label you want to preview. Possible values are 0, 1 and 2. 0 means header label, 1 means normal label and 2 means footer label.`

`LabelSide`

`With this parameter you specify which side of label you want to preview. Possible values are 0 and 1. 0 means front side of the label and 2 means the back side of the label.`

`HasBorder`

`If you specify this parameter the preview will have a border around the preview.`

GetPrintParam

Description:

GetPrintParam function returns the value for provided printing parameter. Possible values for Name parameter are:

SPEED

Function will return the speed parameter as string value.

DARKNESS

Function will return the darkness parameter as string value.

Syntax:

```
Function GetPrintParam(Name As String) As String
```

Explanation of the parameters:

```
Name
Defines which parameter should be retrieved :
PRINTSPEED – speed parameter as string value
PRINTDARKNESS - darkness parameter as string value
PRINTROTATION - rotation parameter as string value
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 - darkness on specific print head on the printer
                 with multiple print heads
NUMBER_OF_FRONT_PAGE_COLORS - number of colors for the specified
                              page side for the printers
                              with multiple print heads
NUMBER_OF_BACK_PAGE_COLORS - number of colors for the specified
                             page side for the printers
                             with multiple print heads
```

See also:

```
Print, PrintePort, PrinterName, SetPrintParam
```

GetPrintParamEX

Description:

This function retrieve the value of printing parameter.

Explanation of the parameters:

```
PrinterName
Defines for which printer you want to retrieve the parameter,
blank value means that default printer will be taken.
ParamName
Defines which parameter should be retrieved :
PRINTSPEED – speed parameter as string value
PRINTDARKNESS - darkness parameter as string value
PRINTROTATION - rotation parameter as string value
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 - darkness on specific print head on the printer
                 with multiple print heads
NUMBER_OF_FRONT_PAGE_COLORS - number of colors for the specified
                              page side for the printers
                              with multiple print heads
NUMBER_OF_BACK_PAGE_COLORS - number of colors for the specified
                             page side for the printers
                             with multiple print heads
```

Syntax:

```
Function GetPrintParamEx(PrinterName As String, ParamName As
String) As String
```

See also:

GetPrinterSpeedList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GetPrintParamList

Description:

This method returns comma separated list of values/names for specified ParamName and specified PrinterName and the data you desire

Explanation of the parameters:

```
PrinterName
Defines for which printer you want to retrieve the parameter,
blank value means that default printer will be taken.
ParamName
It can be one of :
PRINTSPEED – speed parameter as string value
PRINTDARKNESS - darkness parameter as string value
PRINTROTATION - rotation parameter as string value
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 - darkness on specific print head on the printer
                 with multiple print heads
FRONT_PAGE_COLOR_LIST - comma separated list of color values represented
                        in integers(COLORREF)
BACK_PAGE_COLOR_LIST - comma separated list of color values represented
                       in integers(COLORREF)

DataType
Parameter is used for parameters PRINTSPEED, PRINTDARKNESS,
PRINTDARKNESS1, PRINTDARKNESS2, PRINTDARKNESS3, PRINTDARKNESS4,
PRINTDARKNESS5 and it means whether you want to retrieve values list or
names list for the specified parameter. Values are actual values for the
speed/darkness while names are string representations for those items
(for example you use names when displaying this data to user)
```

Syntax:

```
Function GetPrintParamList(PrinterName As String, ParamName As
String, DataType As Long) As String
```

See also:

GetPrinterSpeedList, GetPrinterSpeedUnit, GetPrintParam, PrinterName, PrinterPort

GlobalVariables

Description:

Returns the interface to the variable list for global variables.

Syntax:

```
Property GlobalVariables As IVariableList
```

Access Rights:

read-only

See also:

```
DeleteVariable, NewVariable, SelectVariable, Variables,
VariableProperties, NewVariableWiz
```

IsLabelReadOnly

Description:

IsLabelReadOnly property returns TRUE if label is opened in read only mode, otherwise return value is FALSE.

Syntax:

```
Property IsLabelReadOnly As Boolean
```

Access Rights:

```
read-only
```

IsLinkedToStock

Description:

IsLinkedToStock property returns TRUE if label is linked with some stock or FALSE if none of the stocks are used.

Syntax:

```
Property IsLinkedToStock As Boolean
```

Access Rights:

```
read-only
```

IsStoreRecallPrintingAvailable

Description:

If store recall printing is avaliable value of the IsStoreRecallPrintingAvaliable property is TRUE.

Syntax:

```
Property IsStoreRecallPrintingAvailable As Boolean
```

Access Rights:

```
read-only
```

IsStoreRecallSupported

Description:

If current printer supports store/recall value of the IsStoreRecallSupported property is TRUE.

Syntax:

```
Property IsStoreRecallSupported As Boolean
```

Access Rights:

```
read-only
```

LabelFileName

Description:

LabelFileName represents the name of the label file.

Syntax:

```
LabelFileName As String
```

LabelSettings

Description:

Returns an interface to IlabelSettings class. See more in the Ilabel Chapter.

Syntax:

```
Property LabelSettings As ILabelSettings
```

Access Rights:

```
read-only
```

LastRevisionID

Description:

Returns the ID of the last revision, which was made to the label.

Syntax:

```
Property LastRevisionID As Long
```

Access Rights:

```
read-only
```

NewDatabase

Description:

This method creates a new database access on the label.

Syntax:

```
Function NewDatabase() As Boolean
```

See also:

`Databases, DatabaseProperties, DeleteDatabase, NewDatabaseWiz, SelectDatabase`

NewDatabaseWiz

Description:

NewDatabaseWiz method opens Database Wizard. In case of success method returns value TRUE.

Syntax:

```
Function NewDatabaseWiz() As Boolean
```

See also:

`Databases, DatabaseProperties, DeleteDatabase, NewDatabase, SelectDatabase`

NewFunction

Description:

This method creates a new function on the label.

Syntax:

```
Function NewFunction() As Boolean
```

See also:

`DeleteFunction, Functions, FunctionProperties`

NewVariable

Description:

This method creates a new variable on the label.

Syntax:

```
Function NewVariable() As Boolean
```

See also:

`DeleteVariable, GlobalVariables, SelectVariable, Variables, VariableProperties, NewVariableWiz`

NewVariableWiz

Description:

NewVariableWiz method opens Variable Wizard. In case of success method returns value TRUE.

Syntax:

```
Function NewVariableWiz() As Boolean
```

See also:

`DeleteVariable, GlobalVariables, SelectVariable, Variables, VariableProperties, NewVariable`

Objects

Description:

Returns the interface to the object list.

Syntax:

```
Property Objects As IObjectList
```

Access Rights:

read-only

See also:

`DeleteObject, ObjectPropeties`

ObjectProperties

Description:

ObjectProperties method opens a dialog box with properties for the object selected with ID.

Syntax:

```
Function ObjectProperties(ID As Long) As Boolean
```

See also:

```
DeleteObject, Objects
```

PrintAndGetJobID

Description:

When printing from NiceLabel user can retrieve an exact status of the print job.
Parameter Quantity : Specifies the quantity that should be used when printing
AdditionalParam: Reserved for future use, must be "
Return Value : Job ID of the print Job.
In case of error, return value can contain the following strings:
"Logging Disabled" - Logging is not enabled, retrieving of ID not possible.
"Print Failed" - Print Job failed to print.

Syntax:

Function PrintAndGetJobID(Quantity As String, AdditionalParams As String) As String

Print

Description:

Print method prints as many labels as selected in Quantity.

The quantity parameter is the print quantity of the labels; it can be a number, or one of following words:

- VARIABLE
- UNLIMITED

The first parameter means printing on the base of the variable quantity (one of the variables sets the quantity), the second one means unlimited printing (printing from the whole database file for example).

Labels are printed to the printer which is selected in PrinterName property.

If PrintType is:

- Simulate (value = "Simulate") : the whole printing process will be just simulated. All variable values, functions and database access will be performed, but no actual printing will be done. This parameter can be used to perform a printing validation (validation of the data) prior to real printing.

- NormalPrinting (value = " NormalPrinting ") : normal printing is selected.

- RecallPrinting (value = " RecallPrinting ") : recall printing is selected.

If Location is:

- Stopupdate (value = " Stopupdate ") : he contens of the objects stay unchanged after the print or preview so that the real properties of the object can be retrieved

.- StoreLocation (value = " StoreLocation=XXX") : XXX is slot name. Avaliable slot names can be obtained by calling GetDownloadLoactionsList or GetDownloadLoactionsListForPrinter..

Syntax:

```
Function Print(Quantity [, SkipLabels [, Identical [, Sets]] [,
PrintType] [, Location]]As String) As Boolean
```

Example ***:*** `Print("1,0,2,1")`

```
        Result of print is 2 identical labels.
```

See also:

`GetPrintParam, PrintePort, PrinterName, SetPrintParam`

PrinterName

Description:

PrinterName represents the name of the printer to which label will be printed.

Syntax:

```
PrinterName As String
```

See also:

`GetPrintParam, Print, PrinterPort, SetPrintParam`

PrinterPort

Description:

Command PrinterPort allows you to set different printer port than specified in the label file.

Syntax:

```
PrinterPort As String
```

Access Rights:

`read-write`

See also:

`GetPrintParam, Print, PrinterName, SetPrintParam`

PrintType

Description:

Label mode can be obtained by calling PrintType property
O = PRINT_TYPE_NORMAL
1 = PRINT_TYPE_RECALL

Syntax:

```
PrintType As Long
```

Access Rights:

```
Read
```

Save

Description:

Save method saves the selected label. Please note that Save command is not supported in NiceLabel Engine edition of NiceLabel.

Syntax:

```
Function Save() As Boolean
```

SavedPrinterName

Description:

Returns the printer name that was used when the label was saved, It remains unchanged until next load of the label. If the printer for label is changed and the label is saved, SavedPrinterName property will hold the new printer name after the next load.

Syntax:

```
Property SavedPrinterName As String
```

Access Rights:

```
read-only
```

SelectDatabase

Description:

SelectDatabase method selects a variable on the label. In case of success method returns value TRUE.

Syntax:

```
Function SelectDatabase(ID As Long) As Boolean
```

See also:

`Databases, DatabaseProperties, DeleteDatabase, NewDatabase, NewDatabaseWiz`

SelectVariable

Description:

SelectVariable method selects a variable on the label. In case of success method returns value TRUE.

Syntax:

```
Function SelectVariable(ID As Long) As Boolean
```

See also:

`DeleteVariable, GlobalVariables, NewVariableWiz, Variables, VariableProperties, NewVariable`

SessionEnd

Description:

The function closes data stream.

Syntax:

```
Function SessionEnd() As Boolean
```

See also:

SessionStart

SessionPrint

SessionPrint

Description:

You send the data stream to printer using this function. You can use multiple SessionPrint commands one after another and join them in single data stream. The stream is not closed until the command SessionEnd occurs.

Syntax:

```
Function SessionPrint(Quantity As String) As Boolean
```

See also:

SessionStart

SessionEnd

SessionStart

Description:

All three functions (SessionStart, SessionPrint, SessionEnd) are used together. If ordinary command SessionPrint is used, every time a complete data stream for printer is sent. If you want to join multiple Print commands into one data stream, you can use the command SessionStart followed with any number of SessionPrint and in the end use the command SessionEnd. The stream is not closed until the command SessionEnd occurs. These commands offer a way of optimal printing through NiceCommands and it is not necessary to generate a complete data stream for each print session.

Syntax:

```
Function SessionStart() As Boolean
```

See also:

SessionPrint

SessionEnd

SetPrintJobName

Description:

The INiceLabel interface is extended to support setting the PrintJobName for next print command. The command is defined as:

Parameter JobName must contain the name of the print job. function returns true if successful.

The parameter set is used only for the first print command. After that it is reset to use default print job names. Application should set the print job name each time before executing print commands.

Syntax:

```
SetPrintJobName(JobName As String)
```

SetPrintParam

Description:

With SetPrintParam function it is possible to set the following printing parameters:

- SPEED – Name parameter : PRINTSPEED
- DARKNESS – Name parameter : PRINTDARKNESS
- PAPER BIN – Name parameter : PAPERBIN
- ROTATION – Name parameter : PRINTROTATION
  Value : 0 - no rotation
  Value : 1 - 180° rotation

- LEFT OFFSET FOR ALL PRINTING OBJECTS – Name Parameter: PRINTOFFSETX
  Value: Numeric positive or numeric negative in pixels
- TOP OFFSET FOR ALL PRINTING OBJECTS – Name Parameter: PRINTOFFSETY
  Value: Numeric positive or numeric negative in pixels

The name of the printing parameter should be stored in the function's Name parameter and the value of the parameter in the Value parameter. Both parameters should be represented as a string. If function is able to set the required printing parameters, the return value will be TRUE otherwise something was wrong and the return result will be FALSE.

Syntax:

```
Function SetPrintParam(Name As String, Value As String) As Boolean
```

See also:

```
GetPrintParam, Print, PrinterPort, PrinterName
```

<u>SetDirty</u>

Description:
With this method you can set the dirty flag of the label document to true, if you do this means that the label has been modifed and the user will be prompted to save the label when he will try to close the label

Syntax:
```
Function SetDirty(flag As Boolean)
```

<u>SetObjectVisible</u>

Description:
With this method you can set object visibility on the label.
When the method succeeds, the return value is true. If it fails, the return value is false. The reasons for failing can be:
- Object with specified name is not found (Error 1610).
- Object is in error state. If object is found on the label, but it is in error state (not OK or Phantom), the visible property can not be changed (Error 1611)

Syntax:
```
Function SetObjectVisible(ObjectName As String, Visible As
Boolean) As Boolean
```

Parameters:
Objectname
Name of the object, for which the visible property should be changed

Visible
Visible status after the call. If parameter is false, the object will get status Phantom

StockName

Description:

Returns the name of the stock, which is used on the label.

Syntax:

```
Property StockName As String
```

Access Rights:

```
read-only
```

See also:

```
StockType
```

StockType

Description:

Returns the type of the stock, which is used on the label.

Syntax:

```
Property StockType As String
```

Access Rights:

```
read-only
```

See also:

```
StockName
```

SynchronizedPrinting

Description:

This property defines wether the synchronized printing should be used or not. If you try to set this property to true when the current printer does not support this then the property will be false and error will be set in NiceLabel.

Syntax:

```
Property SynchronizedPrinting As Boolean
```

Variables

Description:

Returns the interface to the variable list for input variables.

Syntax:

```
Property Variables As IVariableList
```

Access Rights:

read-only

See also:

```
DeleteVariable, GlobalVariables, NewVariableWiz, SelectVariable,
VariableProperties, NewVariable
```

VariableProperties

Description:

VariableProperties method opens a dialog box with properties for the variable selected with ID.

Syntax:

```
Function VariableProperties(ID As Long) As Boolean
```

See also:

```
DeleteVariable, GlobalVariables, NewVariableWiz, SelectVariable,
Variables, NewVariable
```

2.9.16 Class IObject (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5
SetVariable	3,4,5	AnchorPoint	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsLocked	3,4,5
		Kind	3,4,5
		Left	3,4,5
		Name	3,4,5
		PageNumber	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5

		Rotation	3,4,5
		Selected	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values are:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right

6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

`Height, Left, Top, Width`

<u>FormatID</u>

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Values	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

<u>Height</u>

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Left, Top, Width`

<u>ID</u>

Description:

ID of the element

Syntax:

```
Property ID As Long
```

<u>IsLocked</u>

Description:

When the element's position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

<u>Kind</u>

Description:

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject

313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

<u>Left</u>

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Height, Top, Width`

<u>Move</u>

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

<u>Name</u>

Description:

Name property represents the name of the object.

Syntax:

```
Name As String
```

<u>PageNumber</u>

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

Access Rights:

read-only

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

`ResizeFlag`

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction

0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

`Resize`

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

`Rotation`

Rotation

Description:

Specifies the element's rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the Rotation property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
RotateFlag
```

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetVariable

Description:

With this method you can set an object to be fixed or assign it a variable, if ID is -1 then the object will be set to be fixed, if the ID is the ID of the variable on this label then the object will use this variable as a data source. If the variable will be successufly set to this object then the return value will be 0 if not then return value will be -1

Syntax:

```
Function SetVariable(ID As Long) As Long
```

Status

Description:

Status of the object.

Possible values are:

Value	Description

0	OK
1	Error condition – object out of label boundary
2	Phantom - non-printable object

Syntax:

```
Property Status As Long
```

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Height, Left, Width`

Variable

Description:

Returns the interface to the variable, which is connected to the OLE object.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Height, Left, Top`

<u>ZOrder</u>

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

`ZOrder As Long`

2.9.17 Class IObjectList (Advanced Only)

Free	Version Availability	Properties	Version Availability
Item	3,4,5	Count	3,4,5

<u>Count</u>

Description:

Count property returns the number of objects, which are defined on the label.

Syntax:

`Property Count As Long`

Access Rights:

`read-only`

<u>Item</u>

Description:

Returns the interface to the object. Object is selected with Index.

Syntax:

`Function Item(Index As Long) As IObject`

2.9.18 Class IOleObject (Advanced Only)

Methods	Version Availability	Properties	Version Availability

Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5
SetVariable	3,4,5	AnchorPoint	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsLocked	3,4,5
		Kind	3,4,5
		Left	3,4,5
		Name	3,4,5
		PageNumber	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		Selected	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

```
Height, Left, Top, Width
```

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date

7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Left, Top, Width

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

IsLocked

Description:

When the element’s position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

Kind

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Top, Width

Name

Description:

Name property represents the name of the OLE object.

Syntax:

```
Name As String
```

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

PageNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

ResizeFlag

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

Resize

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

Rotation

Rotation

Description:

Specifies the element’s rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

RotateFlag

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Left, Width

Variable

Description:

Returns the interface to the variable, which is connected to the OLE object.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Left, Top

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

2.9.19Class IParagraph (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5
SetContents	3,4,5	AnchorPoint	3,4,5
SetVariable	3,4,5	BestFit	3,4,5
		Color	3,4,5
		Contents	3,4,5
		ContentsMask	3,4,5
		FontName	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsBold	3,4,5

		IsInverse	3,4,5
		IsItalic	3,4,5
		IsLocked	3,4,5
		IsMirror	3,4,5
		IsStacked	3,4,5
		IsStrikeOut	3,4,5
		IsUnderlined	3,4,5
		Justification	4,5
		Kind	3,4,5
		Left	3,4,5
		MaskCharacter	4,5
		MaxSizeX	3,4,5
		MaxSizeY	3,4,5
		MinSizeX	3,4,5
		MinSizeY	3,4,5
		NumberOfLines	5
		Name	3,4,5
		PageNumber	3,4,5
		PtSizeX	3,4,5
		PtSizeY	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		Selected	3,4,5
		SpacingX	3,4,5
		SpacingY	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

Height, Left, Top, Width

BestFit

Description:

This property is TRUE if BestFit option for font resizing is enabled.

Syntax:

```
BestFit As Boolean
```

See also:

MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingX, SpacingY

Color

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
Color As Long
```

See also:

`Fontname, IsBold, IsInverse, IsItalic, PtSizeX, PtSizeY`

Contents

Description:

Returns the current contents of the element.

Syntax:

```
Property Contents As String
```

Access Rights:

read-only

ContentsMask

Description:

Sets and gets the ContentsMask attribute.

Syntax:

```
ContentsMask As String
```

FontName

Description:

FontName property represents the name of the font used within paragraph.

Syntax:

```
FontName As String
```

See also:

`Color, IsBold, IsInverse, IsItalic, PtSizeX, PtSizeY`

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Left, Top, Width

<u>ID</u>

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

<u>IsBold</u>

Description:

When the font for the Paragraph element has enabled Bold property then this property is TRUE.

Syntax:

```
IsBold As Boolean
```

See also:

`Color, Fontname, IsInverse, IsItalic, PtSizeX, PtSizeY`

<u>IsInverse</u>

Description:

This property is TRUE if Inverse efect is enabled.

Syntax:

```
IsInverse As Boolean
```

See also:

`Color, Fontname, IsBold, IsItalic, PtSizeX, PtSizeY`

IsItalic

Description:

When the font for the Paragraph element has enabled Italic property then this property is TRUE.

Syntax:

```
IsItalic As Boolean
```

See also:

```
Color, Fontname, IsBold, IsInverse, PtSizeX, PtSizeY
```

IsMirror

Description:

This property is TRUE if Mirror efect is enabled.

Syntax:

```
IsMirror As Boolean
```

IsStacked

Description:
The property defines if the text is normal or stacked

Syntax:

```
IsStacked As Boolean
```

IsStrikeOut

Description:
The property defines if the text is striked out

Syntax:

```
IsStrikeOut As Boolean
```

IsUnderLined

Description:
The property defines if the text is underlined

Syntax:

```
IsUnderlined As Boolean
```

IsLocked

Description:

When the element’s position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

Justification

Description:

Justification proprety represents how the contents of the text object will be justified.

Possible values are:

Value	Description
0	None
1	Left
2	Center
3	Right
4	Full

Syntax:

```
Justification As Long
```

Kind

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject

307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Top, Width

MaskCharacter

This property defines the mask character for the contents mask of the text object

Syntax:

```
MaskCharacter as String
```

MaxSizeX

MaxSizeX property represents the maximim size of the font size in horizontal direction which is allowed when BestFit is enabled.

Syntax:

```
MaxSizeX As Long
```

See also:

BestFit, MaxSizeY, MinSizeX, MinSizeY, SpacingX, SpacingY

MaxSizeY

MaxSizeY property represents the maximim size of the font size in vertical direction which is allowed when BestFit is enabled.

Syntax:

```
MaxSizeY As Long
```

See also:

BestFit, MaxSizeX, MinSizeX, MinSizeY, SpacingX, SpacingY

MinSizeX

Description:

MinSizeX property represents the minimum size of the font size in horizontal direction which is allowed when BestFit is enabled.

Syntax:

```
MinSizeX As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeY, SpacingX, SpacingY

MinSizeY

Description

MinSizeY property represents the minimum size of the font size in vertical direction which is allowed when BestFit is enabled.

Syntax:

```
MinSizeY As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeX, SpacingX, SpacingY

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

NumberOfLines

Description:

This property returns the number of lines in text object for the current contents.

Syntax:

```
NumberOfLines As Long
```

Access Rights:

read-only

Name

Description:

Name property represents the name of the paragraph..

Syntax:

```
Name As String
```

PaheNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

PtSizeX

Description:

PtSizeX represents font size in horizontal direction. Default size is 0. In this case default size for selected font is set.

Syntax:

```
PtSizeX As Long
```

See also:

```
Color, Fontname, IsBold, IsItalic, IsInverse, PtSizeY
```

PtSizeY

Description:

PtSizeY represents vertical font size.

Syntax:

```
PtSizeY As Long
```

See also:

```
Color, Fontname, IsBold, IsItalic, IsInverse, PtSizeX
```

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

```
ResizeFlag
```

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

Resize

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

Rotation

Rotation

Description:

Specifies the element’s rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
RotateFlag
```

<u>Selected</u>

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

<u>SetContents</u>

Description:

When the contents of an element should be changed, SetContents method should be called. In case of success (the Value is valid for the element), the function returns 0. In case of an error, the function returns –1.

Syntax:

```
Function SetContents(Value As String) As Long
```

<u>SetVariable</u>

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

<u>SpacingX</u>

Description:

With SpacingX property you select spacing between each character.

Syntax:

```
SpacingX As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingY

SpacingY

Description:

With SpacingY property you select spacing between each line.

Syntax:

```
SpacingY As Long
```

See also:

`BestFit, MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingX`

Status

Description:

Status of the object.

Possible values are:

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Left, Width

Variable

Description:

Returns the interface to the variable, which is connected to the paragraph.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

AnchorPoint, Height, Left, Top

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
PageNumber
```

2.9.20 Class IParameterList (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Create	3,4	Count	3,4
Item	3,4		

Item

Description:

Returns the interface to the parameter. Function is selected with Index.

Syntax:

```
Function Item(Index As Long) As IDBParameter
```

Count

Description:

Count property returns the number of parameters (filters) defined for a database.

Syntax:

```
Property Count As Long
```

Access Rights:

```
read-only
```

Create

Description:

Create method creates a new filter on the existing database access. Input parameter is a name of the field on which filtering will be executed. Outputi is an interface to the new created filter.

Syntax:

```
Function Create(Field As String) As IDBParameter
```

2.9.21 Class IRectangle (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5

SetVariable	3,4,5	AnchorPoint	3,4,5
		Color	4,5
		FillColor	4,5
		FillStyle	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsLocked	3,4,5
		IsRounded	3,4,5
		Kind	3,4,5
		Left	3,4,5
		LineStyle	3,4,5
		Name	3,4,5
		PageNumber	3,4,5
		PrintAsGraphics	3,4,5
		Radius	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		Selected	3,4,5
		Status	3,4,5
		ThicknessX	3,4,5
		ThicknessY	3,4,5
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

```
Height, Left, Top, Width
```

Color

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
Color As Long
```

See also:

`FillColor, FillStyle, LineStyle, ThicknessX, ThicknessY`

FillColor

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
FillColor As Long
```

See also:

`Color, FillStyle, LineStyle, ThicknessX, ThicknessY`

FillStyle

Description:

FillStyle property represents the fill sytle for the rectangle.

Possible values are:

Value	Description
0	None
1	Clear
2	Solid
3	Right diagonal
4	Left diagonal

8	Vertical
16	Horizontal
32	Cross
64	Cross diagonal
128	25 % of color
256	50 % of color
512	75 % of color

Syntax:

```
FillStyle As Long
```

See also:

`Color, FillColor, LineStyle, ThicknessX, ThicknessY`

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Left, Top, Width`

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

IsLocked

Description:

When the element’s position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

IsRounded

Description:

When this property is set to TRUE, the rectangle be rounded with radius specified in Radius property.

Syntax:

```
IsRounded As Boolean
```

Kind

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Height, Top, Width`

LineStyle

Description:

LineStyle property represents the line sytle for the rectangle.

Possible values are:

Value	Description
0	None
1	Solid
2	Dash
4	Dot
8	Dash + Dot
16	Datsh + Dot + Dot
32	Clear

Syntax:

```
LineStyle As Long
```

See also:

`Color, FillColor, FillStyle, ThicknessX, ThicknessY`

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

Name

Description:

Name property represents the name of the rectangle.

Syntax:

```
Name As String
```

PageNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

```
PageNumber As Long
```

PrintAsGraphics

Description:

When this property is set to TRUE, the rectangle will always be printed as graphics, even if the printer supports printing it with an internal command.

Syntax:

```
PrintAsGraphics As Boolean
```

Radius

Description:

Radius property represents the radius of the rectangle corners.

Syntax:

```
Radius As Long
```

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

```
ResizeFlag
```

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction

0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

Resize

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

```
read-only
```

See also:

```
Rotation
```

Rotation

Description:

Specifies the element’s rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
RotateFlag
```

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

ThicknessX

Description:

`ThicknessX property represents the thickness of the horizontal lines of the rectangle.`

Syntax:

```
ThicknessX As Long
```

See also:

`Color, FillColor, FillStyle, LineStyle, ThicknessY`

ThicknessY

Description:

ThicknessY property represents the thickness of the vertical lines of the rectangle.

Syntax:

```
ThicknessY As Long
```

See also:

`Color, FillColor, FillStyle, LineStyle, ThicknessX`

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

`AnchorPoint, Height, Left, Width`

Variable

Description:

Returns the interface to the variable, which is attached to the element.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

<u>Width</u>

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

```
AnchorPoint, Height, Left, Top
```

<u>ZOrder</u>

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

2.9.22Class IRTFText (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5
SetContents	3,4,5	AnchorPoint	3,4,5
SetVariable	3,4,5	BestFit	3,4,5
		Color	4,5
		Contents	3,4,5
		ContentsMask	3,4,5

		FontName	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsBold	3,4,5
		IsInverse	3,4,5
		IsItalic	3,4,5
		IsLocked	3,4,5
		IsMirror	3,4,5
		IsStacked	3,4,5
		IsStrikeOut	3,4,5
		IsUnderlined	3,4,5
		Justification	4,5
		Kind	3,4,5
		Left	3,4,5
		MaskCharacter	4,5
		MaxSizeX	3,4,5
		MaxSizeY	3,4,5
		MinSizeX	3,4,5
		MinSizeY	3,4,5
		Name	3,4,5
		PageNumber	3,4,5
		PtSizeX	3,4,5
		PtSizeY	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		ScaleFactor	3,4,5
		Selected	3,4,5
		SpacingX	3,4,5
		SpacingY	3,4,5
		Status	3,4,5
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

AnchorElementID

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

```
Width, Height, Left, Top
```

BestFit

Description:

This property is TRUE if BestFit option for font resizing is enabled.

Syntax:

```
BestFit As Boolean
```

Color

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
Color As Long
```

Contents

Description:

Returns the current contents of the element.

Syntax:

```
Property Contents As String
```

Access Rights:

read-only

ContentsMask

Description:

Sets and gets the ContentsMask attribute.

Syntax:

```
ContentsMask As String
```

FontName

Description:

FontName property represents the name of the font used within paragraph.

Syntax:

```
FontName As String
```

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

`Width, AnchorPoint, Left, Top`

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

IsBold

Description:

When the font for the Paragraph element has enabled Bold property then this property is TRUE.

Syntax:

```
IsBold As Boolean
```

IsInverse

Description:

This property is TRUE if Inverse efect is enabled.

Syntax:

```
IsInverse As Boolean
```

IsItalic

Description:

When the font for the Paragraph element has enabled Italic property then this property is TRUE.

Syntax:

```
IsItalic As Boolean
```

IsLocked

Description:

When the element’s position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

IsMirror

Description:

This property is TRUE if Mirror efect is enabled.

Syntax:

```
IsMirror As Boolean
```

IsStacked

Description:
The property defines if the text is normal or stacked

Syntax:

```
IsStacked As Boolean
```

IsStrikeOut

Description:
The property defines if the text is striked out

Syntax:

```
IsStrikeOut As Boolean
```

IsUnderLined

Description:
The property defines if the text is underlined

Syntax:

```
IsUnderlined As Boolean
```

Justification

Description:

Justification proprety represents how the contents of the text object will be justified.

Possible values are:

Value	Description
0	None
1	Right
2	Left
3	Center
4	Full

Syntax:

```
Justification As Long
```

<u>Kind</u>

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

```
Width, AnchorPoint, Hight, Top
```

MaskCharacter

This property defines the mask character for the contents mask of the text object

Syntax:

```
MaskCharacter as String
```

MaxSizeX

MaxSizeX property represents the maximim size of the font size in horizontal direction which is allowed when BestFit is enabled.

Syntax:

```
MaxSizeX As Long
```

See also:

BestFit, MaxSizeY, MinSizeX, MinSizeY, SpacingX, SpacingY

MaxSizeY

MaxSizeY property represents the maximim size of the font size in vertical direction which is allowed when BestFit is enabled.

Syntax:

```
MaxSizeY As Long
```

See also:

BestFit, MaxSizeX, MinSizeX, MinSizeY, SpacingX, SpacingY

MinSizeX

Description:

MinSizeX property represents the minimum size of the font size in horizontal direction which is allowed when BestFit is enabled.

Syntax:

```
MinSizeX As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeY, SpacingX, SpacingY

MinSizeY

Description

MinSizeY property represents the minimum size of the font size in vertical direction which is allowed when BestFit is enabled.

Syntax:

```
MinSizeY As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeX, SpacingX, SpacingY

Move

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

Name

Description:

Name property represents the name of the RTF text.

Syntax:

```
Name As String
```

PageNumber

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

PageNumber As Long

PtSizeX

Description:

PtSizeX represents font size in horizontal direction. Default size is 0. In this case default size for selected font is set.

Syntax:

```
PtSizeX As Long
```

See also:

`Color, Fontname, IsBold, IsItalic, IsInverse, PtSizeY`

PtSizeY

Description:

PtSizeY represents vertical font size.

Syntax:

```
PtSizeY As Long
```

See also:

`Color, Fontname, IsBold, IsItalic, IsInverse, PtSizeX`

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long)
```

See also:

`ResizeFlag`

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

`Resize`

<u>RotateFlag</u>

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

`Rotation`

Rotation

Description:

Specifies the element's rotation. When the element can be rotated only in steps of 90 degrees.

The valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
RotateFlag
```

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetContents

Description:

When the contents of an element should be changed, SetContents method should be called. In case of success (the Value is valid for the element), the function returns 0. In case of an error, the function returns –1.

Syntax:

```
Function SetContents(Value As String) As Long
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

SpacingX

Description:

With SpacingX property you select spacing between each character.

Syntax:

```
SpacingX As Long
```

See also:

BestFit, MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingY

SpacingY

Description:

With SpacingY property you select spacing between each line.

Syntax:

```
SpacingY As Long
```

See also:

```
BestFit, MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingX
```

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition

2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

`Width, AnchorPoint, Left, Height`

Variable

Description:

Returns the interface to the variable, which is attached to the element.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

`Top, AnchorPoint, Left, Height`

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

2.9.23 Class IText (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Move	3,4,5	AnchorElementID	3,4,5
Resize	3,4,5	AnchorLevel	3,4,5
SetContents	3,4,5	AnchorPoint	3,4,5
SetVariable	3,4,5	BestFit	3,4,5
		CharSet	3,4,5
		Color	4,5
		Contents	3,4,5
		ContentsMask	3,4,5
		FontName	3,4,5
		FormatID	3,4,5
		Height	3,4,5
		ID	3,4,5
		IsBold	3,4,5
		IsInverse	3,4,5
		IsItalic	3,4,5
		IsLocked	3,4,5
		IsMirror	3,4,5
		IsRTL	3,4,5
		IsStacked	3,4,5

		IsStrikeOut	3,4,5
		IsUnderlined	3,4,5
		Justification	4,5
		Kind	3,4,5
		Left	3,4,5
		MaskCharacter	4,5
		MaxSizeX	3,4,5
		MaxSizeY	3,4,5
		MinSizeY	3,4,5
		MinSizeY	3,4,5
		NumberOfLines	5
		Name	3,4,5
		PageNumber	3,4,5
		PtSizeX	3,4,5
		PtSizeY	3,4,5
		ResizeFlag	3,4,5
		RotateFlag	3,4,5
		Rotation	3,4,5
		ScaleFactor	3,4,5
		Selected	3,4,5
		SpacingX	3,4,5
		SpacingY	3,4,5
		StartAngle	
		Status	3,4,5
		TextAngle	
		TextDirection	
		Top	3,4,5
		Variable	3,4,5
		Width	3,4,5
		ZOrder	3,4,5

<u>AnchorElementID</u>

Description:

Currently not used. Intended to be used, when position of one element can be dependant on position/size of another element.

Syntax:

```
AnchorElementID As Long
```

AnchorLevel

Description:

Currently not used.

Syntax:

```
AnchorLevel As Long
```

AnchorPoint

Description:

Identifies the point of the object, which is fixed on the label. If the object size is changed, the specified AnchorPoint remains on the same position.

Possible values:

Value	Description
0	top left
1	top center
2	top right
3	mid left
4	mid center
5	mid right
6	bottom left
7	bottom center
8	bottom right

Syntax:

```
AnchorPoint As Long
```

See also:

```
Top, Hight, Left, Height
```

BestFit

Description:

This property is TRUE if BestFit option for font resizing is enabled.

Syntax:

```
BestFit As Boolean
```

See also:

MaxSizeX, MaxSizeY, MinSizeX, MinSizeY, SpacingX, SpacingY

<u>CharSet</u>

Description:

This property sets character set for text element and selected font.

Possible values are:

Language	CharSet
Chinese (Trad.)	136
Croatian	238
Czech	238
Danish	0
Dutch	0
English	0
Finnish	0
French	0
German	0
Greek	161
Hebrew	177
Hungarian	238
Italian	0
Lithuanian	186
Polish	238
Russian	204
Serbian	238
Slovak	238
Slovene	238
Spanish	0
Swedish	0
Turkish	162
Chinese (Simpl.)	134
Japanese	128
Thai	222

Portuguese	0
Ukrainian	204
Norwegian	0

Syntax:

Property CharSet As Long

Access Rights:

```
read-write
```

Color

Description:

With 32-bit value you can specify any RGB color. The color value has the following hexadecimal form:

0x00bbggrr

The low-order byte (rr) contains a value for the relative intensity of red; the second byte (gg) contains a value for green; and the third byte (bb) contains a value for blue. The fourth byte byte must be zero.

Each color parameter can range from 0x0 to 0xFF.

Syntax:

```
Color As Long
```

See also:

```
FontName, IsBold, IsInverse, IsItalic, IsMirror, IsRTL, SpacingX,
SpacingY, PtSizeX, PtSizeY
```

Contents

Description:

Returns the current contents of the element.

Syntax:

```
Property Contents As String
```

Access Rights:

read-only

ContentsMask

Description:

Sets and gets the ContentsMask attribute.

Syntax:

```
ContentsMask As String
```

FontName

Description:

FontName property represents the name of the font used within paragraph.

Syntax:

```
FontName As String
```

See also:

```
Color, IsBold, IsInverse, IsItalic,IsMirror,  IsRTL, SpacingX,
SpacingY, PtSizeX, PtSizeY
```

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

Height

Description:

Height of the element (in 0.01 mm units).

Syntax:

```
Property Height As Long
```

Access Rights:

read-only

See also:

```
Top, AnchorPoint, Left, Width
```

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

IsBold

Description:

When the font for the Paragraph element has enabled Bold property then this property is TRUE.

Syntax:

```
IsBold As Boolean
```

See also:

```
Color, FontName, IsInverse, IsItalic, IsMirror, IsRTL, SpacingX,
SpacingY, PtSizeX, PtSizeY
```

IsInverse

Description:

This property is TRUE if Inverse efect is enabled.

Syntax:

```
IsInverse As Boolean
```

See also:

`Color, FontName, IsBold, IsItalic, IsMirror, IsRTL, SpacingX, SpacingY, PtSizeX, PtSizeY`

IsItalic

Description:

When the font for the Paragraph element has enabled Italic property then this property is TRUE.

Syntax:

```
IsItalic As Boolean
```

See also:

`Color, FontName, IsBold, IsInverse, IsMirror, IsRTL, SpacingX, SpacingY, PtSizeX, PtSizeY`

IsLocked

Description:

When the element’s position is locked on the label, this property has the value TRUE

Syntax:

```
IsLocked As Boolean
```

IsMirror

Description:

This property is TRUE if Mirror efect is enabled.

Syntax:

```
IsMirror As Boolean
```

See also:

`Color, FontName, IsBold, IsInverse, IsItalic, IsRTL, SpacingX, SpacingY, PtSizeX, PtSizeY`

IsRTL

Description:

The client application can check the possibility to set this property by setting it to another value. If the property value (after reading) still has the old value, this means it can not be set. In this case the user interface involved can be disabled.

Syntax:

```
Property IsRTL As Boolean
```

See also:

`Color, FontName, IsBold, IsInverse, IsItalic, IsMirror, SpacingX, SpacingY, PtSizeX, PtSizeY`

IsStacked

Description:

Sets and gets the “Stacked text” attribute.

Syntax:

```
Property IsStacked As Boolean
```

Justification

Description:

Justification proprety represents how the contents of the text object will be justified.

Possible values are:

Value	Description
0	None
1	Right
2	Left
3	Center
4	Full

Syntax:

```
Justification As Long
```

Kind

Element kind.

Possible values are:

Value	Description
301	TextObject
302	RectangleObject
303	BitmapObject
304	BarcodeObject
305	LineObject
306	InverseObject
307	OleObject
308	Downloaded Graphic Object
309	ParagraphObject
312	RTFTextObject
313	EllipseObject

Syntax:

```
Property Kind As Long
```

Access Rights:

read-only

Left

Description:

Left position of the element (in 0.01 mm units).

Syntax:

```
Property Left As Long
```

Access Rights:

read-only

See also:

`Top, AnchorPoint, Width, Height`

MaskCharacter

Description:

With MaskCharacter you can define the mask character for the Text object.

Syntax:

```
MaskCharacter As String
```

<u>**Move**</u>

Description:

Move the element to the location X, Y

Syntax:

```
Move(X As Long, Y As Long)
```

<u>**NumberOfLines**</u>

Description:

This property returns the number of lines in text object for the current contents.

Syntax:

```
NumberOfLines As Long
```

Access Rights:

read-only

<u>**Name**</u>

Description:

Name property represents the name of the RTF text.

Syntax:

```
Name As String
```

<u>**Selected**</u>

<u>**PageNumber**</u>

Description:

The Page index, where the element is located. When Duplex print is used, this index can have the value 0 (first page) or 1 (second page). When duplex printing is not enabled, this property is ignored (the value should always be 0).

Syntax:

```
PageNumber As Long
```

PtSizeX

Description:

PtSizeX represents font size in horizontal direction. Default size is 0. In this case default size for selected font is set.

Syntax:

```
PtSizeX As Long
```

See also:

`Color, FontName, IsBold, IsInverse, IsItalic, IsMirror, SpacingX, SpacingY, IsRTL, PtSizeY`

PtSizeY

Description:

PtSizeY represents vertical font size.

Syntax:

```
PtSizeY As Long
```

See also:

`Color, FontName, IsBold, IsInverse, IsItalic, IsMirror, SpacingX, SpacingY, IsRTL, PtSizeX`

Resize

Description:

Resize the element to the size Width, Height. The element is resized to the closest size in case, that all sizes are not possible.

Syntax:

```
Resize(Width As Long, Height As Long
```

See also:

`ResizeFlag`

ResizeFlag

Description:

Flag, which defines, how the object can be resized.

Possible values are:

Value	Description
0	no resizing
0x88	resizing is possible only in horizontal direction
0x22	resizing is possible only in vertical direction
0x55	resizing is possible only in both directions at the same time

Syntax:

Property ResizeFlag As Long

Access Rights:

read-only

See also:

Resize

RotateFlag

Description:

Flag, which defines, how the object can be rotated.

Possible values are:

Value	Description
0	no rotation
0x07	90 degrees rotation
0x02	180 degrees rotation
0xFF	0-360 degrees rotation in steps of 1 degree

Syntax:

```
Property RotateFlag As Long
```

Access Rights:

read-only

See also:

Rotation

Rotation

Description:

Specifies the element's rotation. When the element can be rotated only in steps of 90 degrees.

Possible valuesThe valid values for the property are:

Value	Description
0	0 deg.
1	90 deg.
2	180 deg.
3	270 deg.

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.

Syntax:

```
Rotation As Long
```

See also:

```
RotateFlag
```

ScaleFactor

Description:

Sets an gets the Scaling factor. Setting this property does not always result in a change - depends on the font type

Syntax:

```
ScaleFactor As Long
```

Selected

Description:

When the element is selected, this property is TRUE.

Syntax:

```
Selected As Boolean
```

SetContents

Description:

When the contents of an element should be changed, SetContents method should be called. In case of success (the Value is valid for the element), the function returns 0. In case of an error, the function returns –1.

Syntax:

```
Function SetContents(Value As String) As Long
```

SetVariable

Description:

Connects the element to the variable with ID. If the return value of the function is –1, then some error occured during the connection. The best example for this is that you want connect variable with fixed length to an element which requires different fixed lenght. (EAN13 barcode). In such case element is not connected to any variable – it is fixed.

Syntax:

```
Function SetVariable(ID As Long) As Long
```

SpacingX

Description:

With SpacingX property you select spacing between each character.

Syntax:

```
SpacingX As Long
```

See also:

```
Color, FontName, IsBold, IsInverse, IsItalic, IsMirror, PtSizeY,
SpacingY, IsRTL, PtSizeX
```

SpacingY

Description:

With SpacingY property you select spacing between each line.

Syntax:

```
SpacingY As Long
```

See also:

```
Color, FontName, IsBold, IsInverse, IsItalic, IsMirror, PtSizeY,
SpacingX, IsRTL, PtSizeX
```

StartAngle

Description:

With StartAngle property you define start text angle of the curved text. Valid values are between 0 and 360 degrees.

Syntax:

```
StartAngle As Long
```

See also:

`TextAngle, TextDirection`

Status

Description:

Status of the object.

Possible values are :

Value	Description
0	OK
1	Error condition
2	Phantom

Syntax:

```
Property Status As Long
```

Access Rights:

read-only

TextAngle

Description:

With TexctAngle property you define max angle of the curved text. Valid values are between 0 and 360 degrees.

Syntax:

```
TextAngle As Long
```

See also:

```
StartAngle, TextDirection
```

TextDirection

Description:

With TexctDirection property you define direction of the curved text. Valid values are:
1 – Clockwise
2 – Counterclockwise

Syntax:

```
TextDirection As Long
```

See also:

```
StartAngle, TextAngle
```

Top

Description:

Top position of the element (in 0.01 mm units).

Syntax:

```
Property Top As Long
```

Access Rights:

read-only

See also:

```
Width, AnchorPoint, Left, Height
```

Variable

Description:

Returns the interface to the variable, which is connected to the text.

Syntax:

```
Property Variable As IVar
```

Access Rights:

read-only

Width

Description:

Width of the element (in 0.01 mm units).

Syntax:

```
Property Width As Long
```

Access Rights:

read-only

See also:

```
Top, AnchorPoint, Left, Height
```

ZOrder

Description:

Value of Z order of the object position. When the Zorder value is the highest, the element is placed on the top of all others.

Syntax:

```
ZOrder As Long
```

2.9.24Class IVar (Advanced Only)

Methods	Version Availability	Properties	Version Availability
DependsOnFunction	3,4,5	AutoTrim	4,5
GetPureValue	4,5	BreakLines	3,4,5
GetValue	3,4,5	Default	3,4,5
SetValue	3,4,5	DefaultValue	3,4,5

ShouldTraceNow	3,4,5	DefType	3,4,5
Undefine	3,4,5	Description	3,4,5
		DynamicValue	3,4,5
		FixedLength	3,4,5
		FormatID	3,4,5
		HasMaxValue	3,4,5
		HasMinValue	3,4,5
		ID	3,4,5
		IncrementCount	3,4,5
		IncrementKind	3,4,5
		IncrementStep	3,4,5
		IncrementStep2	3,4,5
		IncrementStep3	3,4,5
		IncrementType	3,4,5
		InputPicture	3,4,5
		IsBasedOnQuantity	3,4,5
		IsDisabled	3,4,5
		IsInternal	3,4,5
		IsMultiLine	3,4,5
		IsStoreVariable	5
		IsUsed	3,4,5
		IsVarQuantity	3,4,5
		JustificationType	3,4,5
		Length	3,4,5
		LineCount	3,4,5
		LineLength	3,4,5
		MaxValue	3,4,5
		MinValue	3,4,5
		Name	3,4,5
		OutputPicture	3,4,5
		PadCharacter	3,4,5
		PickList	3,4,5
		PictureType	3,4,5
		Prefix	3,4,5
		Prompt	3,4,5
		PromptCount	3,4,5

		PromptOrder	3,4,5
		RollOver	3,4,5
		StrictChecking	3,4,5
		Suffix	3,4,5
		Tag	4,5
		TraceOn	3,4,5
		ValueRequired	3,4,5
		VarType	3,4,5
		WordWrap	3,4,5

AutoTrim

Description:

AutoTrim property defines the behaviour of SetContents function of IVar interface. When AutoTrim is TRUE the value assinged to the variable is first trimmed to the length of the variable - this prevents errors when the provided value is too long.

Default value is false.

Syntax:

```
Propery AutoTrim As Boolean
```

BreakLines

Description:

If variable has BreakLines option enabled then value of the WordWrap property is TRUE.

Syntax:

```
BreakLines As Boolean
```

Default

Description:

Default method returns the current value of the variable.

Syntax:

```
Property Default As String
```

Access Rights:

read-only

DefaultValue

Description:

In DefaultValue property is stored a default value for the variable.

Syntax:

```
DefaultValue As String
```

See also:

```
DynamicValue, FixedLenght, IsBasedonQuantity, ValueRequired
```

DefType

Description:

DefType property represents the case on which default value for the variable will be selected.

Possible values are:

Value	Description
0	None – No default value
1	Prompt – You are prompted for default value
2	No prompt – Default value is stored in Default value property

Syntax:

```
DefType As Long
```

DependsOnFunction

Description:

DependsOnFunction method is used for checking if variable is dependet of the function. For input parameter is used function ID. In case that variable is dependet of the function then return value is TRUE.

Syntax:

```
Function DependsOnFunction(FunctionID As Long) As Boolean
```

Description

Description:

Description property represents the description of the variable.

Syntax:

```
Description As String
```

DynamicValue

Description:

DynamicValue has value TRUE, if "Dynamic value" option for default value is enabled.

Syntax:

```
DynamicValue As Boolean
```

See also:

```
DefaultValue, FixedLenght, IsBasedonQuantity, ValueRequired
```

FixedLength

Description:

If variable has fixed lenght then FixedLength property has value TRUE.

Syntax:

```
FixedLength As Boolean
```

See also:

```
DefaultValue, DynamicValue, IsBasedonQuantity, ValueRequired
```

FormatID

Description:

This is the ID of a contents format, which specifies the character set, which is allowed to be used for the element.

The following IDs are valid:

Value	Description
0	All
1	Numeric
2	Alphanumeric
3	Letters
4	7 bit
5	Hex
6	Date
7	Time

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

Syntax:

```
FormatID As Long
```

GetPureValue

Description:

This function returns a pure value of the variable without any prefix or suffix in case they are defined.

Syntax:

```
Function GetPureValue() As String
```

See also:

`GetValue, SetValue`

GetValue

Description:

GetValue method returns the current value of the variable.

Syntax:

```
Function GetValue() As String
```

See also:

`GetPureValue, SetValue`

HasMaxValue

Description:

HasMaxValue property has value TRUE, if range check option Max value is enabled.

Syntax:

```
HasMaxValue As Boolean
```

See also:

`HasMinValue, MaxValue, MinValue, RollOver`

HasMinValue

Description:

HasMinValue property has value TRUE, if range check option Min value is enabled.

Syntax:

```
HasMinValue As Boolean
```

See also:

`HasMaxValue, MaxValue, MinValue, RollOver`

ID

Description:

ID of the element

Syntax:

```
Property ID As Long
```

Access Rights:

read-only

IncrementCount

Description:

IncrementCount property represents the number on how many printed labels counter will be incremented.

Syntax:

```
IncrementCount As Long
```

See also:

`IncrementKind, IncrementStep, IncrementStep2, IncrementStep3, IncrementType`

IncrementKind

Description:

IncrementKind property represents the kind of the increment. Increment can be performed on several types of variables.

Possible values are:

Value	Description

0	Integer
1	Date
2	Time

Syntax:

```
Property IncrementKind As Long
```

Access Rights:

read-only

See also:

`IncrementCount, IncrementStep, IncrementStep2, IncrementStep3, IncrementType`

IncrementStep

Description:

IncrementStep property represents the step of the increment.

Please Note:

Incrementing depends on the type of the variable. If variable type is Integer then only IncrementStep property is used. For all other types such as Time or Date there are IncrementStep2 and IncrementStep3. In each of them there is only one number.

Date:

Days are stored in IncrementStep

Months are stored in IncrementStep2

Years are stored in IncrementStep3

Time:

Hours are stored in IncrementStep

Minutes are stored in IncrementStep2

Seconds are stored in IncrementStep3

Syntax:

```
IncrementStep As Long
```

See also:

`IncrementCount, IncrementKind, IncrementStep2, IncrementStep3, IncrementType`

IncrementStep2

Description:

See IncrementStep

Syntax:

```
IncrementStep2 As Long
```

See also:

`IncrementCount, IncrementKind, IncrementStep, IncrementStep3, IncrementType`

IncrementStep3

Description:

See IncrementStep

Syntax:

```
IncrementStep3 As Long
```

See also:

`IncrementCount, IncrementKind, IncrementStep, IncrementStep2, IncrementType`

IncrementType

Description:

IncrementType property represents the type of incrementation.

Possible values are:

Value	Description
0	None
1	Increment
2	Decrement
3	Offset

Syntax:

```
Property IncrementType As Long
```

Access Rights:

read-only

See also:

`IncrementCount, IncrementKind, IncrementStep, IncrementStep2, IncrementStep3`

InputPicture

Description:

InputPicture property is represents a string (Input format), which can be set in Variable dialog box under Format tab. Each type of variable has it's own possibilites for input pictures.

Syntax:

```
InputPicture As String
```

IsBasedOnQuantity

Description:

IsBasedOnQuantity property has value TRUE, if "Based on var. quantity" is enabled.

Syntax:

```
IsBasedOnQuantity As Boolean
```

See also:

`DefaultValue, DynamicValue, FixedLenght, ValueRequired`

IsDisabled

Description:

This property sets (or gets) the variables “disabled” attribute. When this attribute is set, the variable is not shown in the variables selection lists.

Syntax:

```
Property IsDisabled As Boolean
```

IsInternal

Description:

ReadOnly property, which tells the client if this is an internal variable – the variable, which gets the value from the system rather than from the label data.

Syntax:

```
Property IsInternal As Boolean
```

Access Rights:

```
read-only
```

IsMultiLine

Description:

If variable has MultiLine option enabled then value of the IsMultiLine property is TRUE.

Syntax:

```
IsMultiLine As Boolean
```

See also:

`IsVarQuantity, Length, PickList`

IsStoreVariable

Description:

If variable is store variable value of the IsStoreVariable property is TRUE.

Syntax:

```
IsStoreVariable As Boolean
```

IsUsed

Description:

IsUsed has value TRUE, if variable is connected with an element on the label. In the other case IsUsed has value FALSE.

Syntax:

```
property IsUsed As Boolean
```

Access Rights:

read-only

IsVarQuantity

Description:

IsVarQuantity property has value TRUE, if "Treat as variable quantity" is enabled.

Syntax:

```
IsVarQuantity As Boolean
```

See also:

```
IsMultiLine, Length, PickList
```

JustificationType

Description:

JustificationType proprety represents how the contents of the variable will be justified.

Possible values are:

Value	Description
0	None
1	Right
2	Left
3	Center
4	Full

For the variables full justification type is not used.

Syntax:

```
JustificationType As Long
```

Length

Description:

Lenght propery represents the lenght of the variable. Length can have value between 1 and 4096.

Syntax:

```
Length As Long
```

See also:

```
IsMultiLine, IsVarQuantity, PickList
```

LineCount

Description:

LineCount property represents the number of lines in one variable. LineCount can have value between 1 and 100.

Syntax:

```
LineCount As Long
```

LineLength

Description:

LineLength property represents the lenght of the one line. LineLength can have value between 1 and 4096.

Syntax:

```
LineLength As Long
```

MaxValue

Description:

MaxValue represents the minimal value for rangechecking.

Syntax:

```
MaxValue As String
```

See also:

`HasMaxValue, HasMinValue, MinValue, RollOver`

MinValue

Description:

MinValue represents the minimal value for rangechecking.

Syntax:

```
MinValue As String
```

See also:

`HasMaxValue, HasMinValue, MaxValue, RollOver`

Name

Description:

Name property represents the name of the variable. Name can be only 12 characters long.

Syntax:

```
Name As String
```

OutputPicture

Description:

OutputPicture property is represents a string (Output format), which can be set in Variable dialog box under Format tab. Each type of variable has it's own possibilites for output pictures.

Syntax:

```
OutputPicture As String
```

PadCharacter

Description:

PadCharacter property represents the PadCharacter of the variable.

Syntax:

```
PadCharacter As String
```

See also:

```
Prefix, Suffix
```

PickList

Description:

This property retrieves the pick list for the current variable.

Syntax:

```
Property PickList As String
```

Access Rights:

```
read-only
```

See also:

```
IsMultiLine, IsVarQuantity, Length
```

PictureType

Description:

Picture type property represents, which picture for the variable is selected.

Possible values are:

Value	Description
0	None

1	Date
2	Time
3	Floating point
4	Money
5	List
6	Binary

Syntax:

```
PictureType As Long
```

Prefix

Description:

Prefix property represents the prefix of the variable.

Syntax:

```
Prefix As String
```

See also:

`PadCharacter, Suffix`

Prompt

Description:

In Prompt property is stored a prompt string.

Syntax:

```
Prompt As String
```

PromptCount

Description:

PromptCount property represents the number on how many printed labels it will be prompted.

Syntax:

```
PromptCount As Long
```

PromptOrder

Description:

ReadWrite property, with which you can get or set the prompt order of the variable.

Syntax:

```
Property PromptOrder As Integer
```

Access Rights:

```
read-write
```

RollOver

Description:

RollOver has value TRUE, if "RollOver" option is enabled.

Syntax:

```
RollOver As Boolean
```

See also:

```
HasMaxValue, HasMinValue, MaxValue, MinValue
```

SetValue

Description:

SetValue method is used for setting the value for the variable.

Syntax:

```
Function SetValue(Value As String) As Long
```

See also:

```
GetPureValue, GetValue
```

ShouldTraceNow

Description:

In case that you want save value of the variable to the log file you should use ShouldTraceNow method.

Syntax:

```
Function ShouldTraceNow(bStart As Boolean) As Boolean
```

StrictChecking

Description:

StrictChecking property has value TRUE, if strict checking flag is enabled. Strict checking flag prevents to connect a variable, with different length that is required by element, to an element.

Syntax:

```
StrictChecking As Boolean
```

Suffix

Description:

Suffix property represents the suffix of the variable.

Syntax:

```
Suffix As String
```

See also:

```
PadCharacter, Prefix
```

Tag

Description:

Tag property is a simple integer property, that the client application can use for any kind of internal processing.

Syntax:

```
Propery Tag As Long
```

TraceOn

Description:

TraceOn has value TRUE, if variable tracing is enabled.

Syntax:

```
TraceOn As Boolean
```

Undefine

Description:

Sets the InputType back to original. Intended Use:

When one database variable is set, its value is never read again from the database until the label is closed. With the Undefine call, the variable gets “original state”. This means that its value will be retrieved from the database although the label is not closed in between.

Syntax:

```
Function Undefine() As Boolean
```

ValueRequired

Description:

ValueRequired has value TRUE, if "ValueRequired" option for prompted variable is enabled.

Syntax:

```
ValueRequired As Boolean
```

See also:

`DefaultValue, DynamicValue, FixedLenght, IsBasedonQuantity`

VarType

Description:

Vartype property represents the type of the variable.

Vartype property can have following values:

Value	Description
1	Prompt
2	System clock
3	Printer clock
4	Global
5	Generated
6	Overload
7	Database
8	Contents provider

Please note:

Overload type of the variable is internal type of the variable. It is used, when external application sets value for the variable.

Contents provider is variable type, which is invisible to the user. Contents provider variable is the result of contents provider.

Syntax:

```
Property VarType As Long
```

Access Rights:

read-only

WordWrap

Description:

If variable has WordWrap option enabled then value of the WordWrap property is TRUE.

Syntax:

```
WordWrap As Boolean
```

2.9.25 Class IVariableList (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Add	3,4,5	Count	3,4,5
Create	3,4,5		
CreateGlobal	5		
CreatePrompted	3,4,5		
Delete	3,4,5		
FindByID	3,4,5		
FindByName	3,4,5		
FlushList	3,4,5		
Item	3,4,5		
Remove	3,4,5		
SetValues	4,5		

Add

Description:

Add method adds a variable with specific ID to the variable list. In case of success TRUE result is returned.

Syntax:

```
Function Add(ID As Long) As Boolean
```

Count

Description:

Count property returns the number of variables, which are defined on the label.

Syntax:

```
Property Count As Long
```

Access Rights:

read-only

Create

Description:

Create method creates new variable. Output result is interface to the new variable and input parameter is the name of the new variable.

Syntax:

```
Function Create(Name As String) As IVar
```

See also:

`Create, CreatePrompted, Delete, Remove`

CreateGlobal

Description:

CreateGlobal method creates new global variable variable. Output result is interface to the new variable and input parameter is the name of the new global variable.

Syntax:

```
Function CreateGlobal(Name As String, Value As String, Format As
Long, Length As Long, Increment As Long) As IVar
```

Explanation of the parameters:

`Name`
`Represents the global variable name.`
`Value`
`Represents current value of the global variable name.`
`Format`
`Represents data format.`

ALL	0
NUMERIC	1
ALPHANUMERIC	2
LETTERS	3
7BIT	4
HEX	5
DATE	6
TIME	7

`Lenght`

`Represents global variable lenght.`

`Incremenet`

```
Represents global variable increment.
    =0  – no incremenent
    >0  - incremenet
    <0  - decrement
```

See also:

`Create, CreatePrompted, Delete, Remove`

CreatePrompted

Description:

Create method creates new prompted variable. Output result is interface to the new variable and input parameter is the name of the new variable.

Syntax:

```
Function CreatePrompted(Name As String) As IVar
```

See also:

`Create, Delete, Remove`

Delete

Description:

Delete method deletes a variable with specific ID. In case of success TRUE result is returned.

Syntax:

```
Function Delete(ID As Long) As Boolean
```

See also:

`Create, CreatePrompted, Remove`

FindByID

Description:

Returns the interface to the existing variable. Variable is selected with ID.

Syntax:

```
Function FindByID(ID As Long) As IVar
```

See also:

`FindByName`

FindByName

Description:

Description:

Returns the interface to the existing variable. Variable is selected with Name.

Syntax:

```
Function FindByName(Name As String) As IVar
```

See also:

`FindByID`

FlushList

Description:

FlushList method erases the whole variable list. If the operation was sucessfull TRUE value is returned.

Syntax:

```
Function FlushList() As Boolean
```

Item

Description:

Returns the interface to the existing variable. Variable is selected with Index.

Syntax:

```
Function Item(Index As Long) As IVar
```

Remove

Description:

Remove method removes a variable with specific ID from the variable list. In case of success TRUE result is returned.

Syntax:

```
Function Remove(ID As Long) As Boolean
```

See also:

`Create, CreatePrompted, Delete`

SetValues

Description:

The IVariableList interface is extended to support setting multiple variable values with a single command. The command is defined as:

Parameter Values should contain multiline value (one line for each variable). Each line must be defined in format:

VariableName=VariableValue

The command will process each line and set the value for the variable. In case the variable name is invalid (variable with that name does not exist),

the function will return an error.

In case the value is not valid for a variable, the function returns an error.

If en error is encountered, the processing of the functon will be terminated (remaining lines will not be processed).

Functon returns 0 if the processing completes successfully. If not, an error code is returned and calling application can use ErrorID and ErrorMessage properties of the niceApp interface to process the error.

Syntax:

```
SetValues(Values As String)
```

2.9.26 Class ILabelSettings (Advanced Only)

Methods	Version Availability	Properties	Version Availability
SetLabelSize	4,5	Author	3,4,5
		DefaultQuantity	4,5
		Description	3,4,5
		GapX	3,4,5
		GapY	3,4,5
		IsMultiband	3,4,5
		LabelDimensions Unit	3,4,5
		LabelOffsetX	3,4,5
		LabelOffsetY	3,4,5
		LabelSizeX	3,4,5
		LabelSizeY	3,4,5
		NumberOfBands	3,4,5
		Overlap	3,4,5
		PageOffsetX	3,4,5
		PageOffsetY	3,4,5
		PageSizeX	3,4,5
		PageSizeY	3,4,5
		PaperFormat	3,4,5
		PaperSizeX	3,4,5
		PaperSizeY	3,4,5
		PhysPageSizeX	3,4,5
		PhysPageSizeY	3,4,5
		PrintPages	3,4,5
		RadiusX	3,4,5
		RadiusY	3,4,5
		Title	3,4,5
		UGapX	3,4,5
		UGapY	3,4,5
		ULabelOffsetB	3,4,5
		ULabelOffsetR	3,4,5
		ULabelOffsetX	3,4,5
		ULabelOffsetY	3,4,5

		ULabelSizeX	3,4,5
		ULabelSizeY	3,4,5
		UOverlap	3,4,5
		UPaperSizeX	3,4,5
		UPaperSizeY	3,4,5
		URadiusX	3,4,5
		URadiusY	3,4,5
		UGapY	3,4,5
		WebX	3,4,5
		WebY	3,4,5

Common to all properties:

All sizes are measured in the unit that is set in LabelDimensionsUnit if the property name has "U" as the first letter of the name.

Author

Description:

Returns the name of the author of the label

Syntax:

Property Author As String

DefaultQuantity

Description:

Returns the default quantity, which is suggested before label production.

Syntax:

Property DefaultQuantity As Long

Access Rights:

read-only

Description

Description:

Returns the description of the label.

Syntax:

Property Description As String

Access Rights:

read-only

GapX

Description:

Returns the gap between the labels in mm/100 for the X axis.

UGapX - gap between the labels in LabelDimensionsUnit (X axis)

Syntax:

Property GapX As Long

Access Rights:

read-only

See also:

`GapY`

GapY

Description:

Returns the gap between the labels in mm/100 for the Y axis.

UGapY - gap between the labels in LabelDimensionsUnit (Y axis)

Syntax:

Property GapY As Long

Access Rights:

read-only

See also:

`GapX`

IsMultiband

Description:

Returns true if the label is desinged for multiband printing, otherwise result of the property is false.

Syntax:

Property IsMultiBand As Boolean

Access Rights:

read-only

See also:

`NumberOfBands`

LabelDimensionsUnit

Description:

Returns the unit of measure used for the label settings.

Syntax:

Property LabelDimensionsUnit As String

Access Rights:

read-only

LabelOffsetX

Description:

Returns the left offset, which is defined in the label.

ULabelOffsetR - right offset (calculated by NiceLabel)

Syntax:

Property LabelOffsetX As Long

Access Rights:

read-only

See also:

`LabelOffsetY`

<u>LabelOffsetY</u>

Description:

Returns the top offset, which is defined in the label.

ULabelOffsetB - bottom offset (calculated by NiceLabel)

Syntax:

Property LabelOffsetY As Long

Access Rights:

read-only

See also:

`LabelOffsetX`

<u>LabelSizeX</u>

Description:

Returns the height of the label.

ULabelSizeX represents the width of the label in defined units.

See also:

`LabelSizeX`

Syntax:

Property LabelSizeX As Long

Access Rights:

read-only

<u>LabelSizeY</u>

Description:

Returns the width of the label.

ULabelSizeY represents the width of the label in defined units.

Syntax:

Property LabelSizeY As Long

Access Rights:

read-only

See also:

`LabelSizeY`

NumberOfBands

Description:

Returns the number of bands, which are used in the label when multi-band label printing is enabled.

Syntax:

Property NumberOfBands As Long

Access Rights:

read-only

See also:

`IsMultiband`

Overlap

Description:

Overlapping of the labels when using multi-band label printing in mm/100

Syntax:

Property Overlap As Long

Access Rights:

read-only

PageOffsetX

Description:

Offset for the unprintable area (X axis)

Syntax:

Property PageOffsetX As Long

Access Rights:

read-only

See also:

`PageOffsetX`

PageOffsetY

Description:

Offset for the unprintable area (Y axis)

Syntax:

Property PageOffsetY As Long

Access Rights:

read-only

See also:

`PageOffsetY`

PageSizeX

Description:

Returns the width of the printable page.

Syntax:

Property PageSizeX As Long

Access Rights:

read-only

See also:

`PageSizeY`

PageSizeY

Description:

Returns the height of the printable page.

Syntax:

Property PageSizeY As Long

Access Rights:

read-only

See also:

`PageSizeX`

PaperFormat

Description:

Returns the format of the paper. This is the dmPaperSize property from the DEVMODE structure of the printer driver.

Syntax:

Property PaperFormat As Long

Access Rights:

read-only

PaperSizeX

Description:

Returns the paper size. This is the dmPaperWidth * 10 property from the DEVMODE structure of the printer driver.

UPaperSizeX - the information expressed in LabelDimensionsUnit

Syntax:

Property PaperSizeX As Long

Access Rights:

read-only

See also:

`PaperSizeY`

PaperSizeY

Description:

Returns the paper size. This is the dmPaperHeight * 10 property from the DEVMODE structure of the printer driver.

UPaperSizeY - the information expressed in LabelDimensionsUnit

Syntax:

Property PaperSizeX As Long

Access Rights:

read-only

See also:

`PaperSizeX`

PhysPageSizeX

Description:

Returns the whole width of the page.

Syntax:

Property PhysPageSizeX As Long

Access Rights:

read-only

See also:

`PhysPageSizeY`

PhysPageSizeY

Description:

Returns the whole height of the page.

Syntax:

Property PhysPageSizeY As Long

Access Rights:

read-only

See also:

`PhysPageSizeX`

PrintPages

Description:

Property returns true if the Print pages option is used in print dialog otherwise it returns false. This option can be selected when printing multiple labels across.

Syntax:

Property RadiusX As Long

Access Rights:

read-only

RadiusX

Description:

Returns the radius value for the X axis of the label.

UradiusX in LabelDimensionsUnit (X axis)

Syntax:

Property RadiusX As Long

Access Rights:

read-only

See also:

`RadiusX`

RadiusY

Description:

Returns the radius value for the Y axis of the label.

UradiusY in LabelDimensionsUnit (Y axis)

Syntax:

Property RadiusY As Long

Access Rights:

read-only

See also:

`RadiusY`

<u>**SetLabelSize**</u>

Description:

Use this method if you want to change the label size and other label settings. What is possible to set can be seen in the parameters.

Syntax:

Function SetLabelSize(LabelWidth As Long, LabelHeight As Long, LeftOffset As Long, TopOffset As Long, RightOffset As Long, BottomOffset As Long) As Boolean

Access Rights:

read-write

<u>**Title**</u>

Description:

Property retrieves the title of the label.

Syntax:

Property Title As String

Access Rights:

read-write

UGapX

Description:
Property retrieves the GapX label setting in users measurment unit.

Syntax:

Property UGapX As String

Access Rights:

read-write

UGapY

Description:
Property retrieves the GapY label setting in users measurment unit.

Syntax:

Property UGapY As String

Access Rights:

read-write

ULabelOffsetB

Description:

Property retrieves bottom label offset setting in users measurment unit.

Syntax:

Property ULabelOffsetB As String

Access Rights:

read-write

ULabelOffsetR

Description:

.

Syntax:

Property ULabelOffsetR As String

Access Rights:

read-write

<u>ULabelOffsetX</u>

Description:

Property retrieves left label offset setting in users measurment unit.

Syntax:

Property ULabelOffsetX As String

Access Rights:

read-write

<u>ULabelOffsetY</u>

Description:

Property retrieves top label offset setting in users measurment unit.

Syntax:

Property ULabelOffsetY As String

Access Rights:

read-write

<u>ULabelOffsetR</u>

Description:

Property retrieves right label offset setting in users measurment unit.

Syntax:

Property ULabelOffsetR As String

Access Rights:

read-write

<u>ULabelSizeX</u>

Description:

Property retrieves label width setting in users measurment unit.

Syntax:

Property ULabelSizeX As String

Access Rights:

read-write

<u>ULabelSizeY</u>

Description:

Property retrieves label height setting in users measurment unit.

Syntax:

Property ULabelSizeY As String

Access Rights:

read-write

<u>UOverlap</u>

Description:

Property retrieves multiband overlaping setting in users measurment unit.

Syntax:

Property UOverlap As String

Access Rights:

read-write

<u>UPaperSizeX</u>

Description:

Property retrieves paper width setting in users measurment unit.

Syntax:

Property UPaperSizeX As String

Access Rights:

read-write

UPaperSizeY

Description:

Property retrieves paper height setting in users measurment unit.

Syntax:

Property UPaperSizeY As String

Access Rights:

read-write

URadiusX

Description:

Property retrieves horizontal label radius setting in users measurment unit.

Syntax:

Property URadiusX As String

Access Rights:

read-write

URadiusY

Description:

Property retrieves vertical label radius setting in users measurment unit.

Syntax:

Property URadiusY As String

Access Rights:

read-write

WebX

Description:

Returns the number of labels on the page (X axis)

Syntax:

Property WebX as Long

Access Rights:

read-only

See also:

`WebY`

WebY

Description:

Returns the number of labels on the page (Y axis)

Syntax:

Property WebY as Long

Access Rights:

read-only

See also:

`WebX`

2.10 NiceLabel ActiveX interfaces for advanced users

In certain cases additional methods and properties can be used by advanced users.

2.10.1Class Application

Methods	Version Availability	Properties	Version Availability
JobRunEx	4,5	ReadWriteOpen	3,4,5
LabelImport	4,5	IsPrintOnly	5
SetControlEvent	4,5		
TestConnection	3,4,5		

IsPrintOnly

Description:
This property returns true if the NiceLabel/NiceEngine is running in print only mode. When in print only mode some activex methods are not working or are working in different manner.

Syntax:

```
IsPrintOnly As Boolean
```

JobRunEx

Description:
With this method NiceLabel is able to read the third-party job file, and execute the printing. NiceLabel will use the Variable import filter DLLs to be able to execute “third party” job files.
This method identifies the DLL based on the JobType parameter. In case the Variable import filter DLL with specified JobType is not registered an error will be produced.
When appropriate filter DLL will be identified, NiceLabel will call the Conversion function in the DLL, which converts the third-party job file to NiceLabel’s native job file format. In order to support UNICODE values of the job file, NiceLabel will be updated to support UTF-8 formatted JOB files (UTF8 boom must be included at the beginning of the JOB file).
For implementation of the method check Oracle XML filter in the Nicelabel.

Syntax;

```
Function JobRunEx(FileName As String, JobType As Long) As Boolean
```

Explanation of the parameters:

FileName
Represents the Job file name.
JobType
Number of Job types that DLL supports

LabelImport

Description:
This method is used to import label files from another labeling software. Import filter has to be design on the user side.
Import filter is a DLL, which is loaded on runtime when user requests to import labels from another labeling software. The DLL is registered in the registry. When used, it passes information about available import file types, which can be imported using the selected filter.
NiceLabel is using the import filter DLL interface to retrieve the available file types that can be used with the filter. After user of NiceLabel browses the file, NiceLabel passes this file name to the filter, and the filter creates NiceLabel’s LBT file format.
LBT is a text-formatted file, which NiceLabel recognizes and can directly load.

Syntax:

```
Function LabelImport(FilterID As Long, FormatID As Long,
ImportFile As String, ExportFile As String, EmbedGraphics As
Boolean) As Boolean
```

Explanation of the parameters:

```
FilterID
Name of the import filter (1,2,...)
FormatID
Name of the import filter function (1,2,...)
ImportFile
Path name to the original file.
ExportFile
Pathname to the destination file.
EmbedGraphics
Background picture embedded in the label:
True:  embedded
False: not embedded
```

ReadWriteOpen

Description:
With this property you can control wether the labels opened through ActiveX are opened in read only mode or not. If this property is true then the label is opened in read/write mode and if modified it can be saved.

Syntax:

```
ReadWriteOpen As Boolean
```

SetControlEvent

Description:
Event objects provide means to synchronize between the caller application and the server application. PrintEvent handling is implemented on Application level. The Event can be set from calling application (ControlEvent). The setting method also defines, if the ControlEvent should be released immediately after the first printing action (creation of print job), or the ControlEvent will be released by calling application (calling the event setting method with NULL parameter).

Syntax:

```
Function SetControlEvent(Event As Unknown, AutoRelease As Boolean)
As Boolean
```

Explanation of the parameters:

```
Event
It pass the Controlling event to NiceLabel application. If this
parameter is NULL, any existing controlling event will be
released.
AutoRelease
TRUE: after the print action is completed, the event will be
```

```
released. Subsequent print events will not have a control event
(new control event should be set from the caller application)
FALSE: after the printing is completed, the control event will not
be released. If new print actions will be called, they will be
logged under the same print event.
```

TestConnection

Description:
Method test if the connection to INiceApp is established.

Syntax:

```
Function TestConnection() As Boolean
```

2.10.2 Class IDatabase

Methods	Version Availability	Properties	Version Availability
Check	4,5	HasFilter	3,4,5
		IsChecked	4,5
		Level	4,5

Check

Description:
If the IsChecked property is set to false then the call to this method performs calculation of the function based on current data and return 0 if successfull or non 0 if not successfull. Also the IsChecked property is set to true after this call.

Syntax:

```
Function Check() As Long
```

HasFilter

Description:
This property returns true if the database definition uses filter.

Syntax:

```
Property HasFilter As Boolean
```

IsChecked

Description:
If this property is set to false then the call to method Check performs calculation of the function based on current data and return 0 if successfull or non 0 if not successfull. Also this property is set to true after this call.

.

Syntax:

```
Property IsChecked As Boolean
```

Level

Description:
If the function is used in some other function then this function will have level 2 while its parent function will have level 1, this goes in depth depending on how many dependant function you have. It is needed when calculating the functions because those functions with higher level need to be calculated first in order that functions with lower level are calculated properly.

Syntax:

```
Property Level As Long
```

Explanation of the parameters:

FilterID
level number

2.10.3Class IEXTFunction (Advanced Only)

Methods	Version Availability	Properties	Version Availability
Check	4,5	IsChecked	4,5
		Level	4,5

Check

Description:
If the IsChecked property is set to false then the call to this method performs calculation of the function based on current data and return 0 if successfull or non 0 if not successfull. Also the IsChecked property is set to true after this call.

Syntax:

```
Function Check() As Long
```

IsChecked

Description:
If this property is set to false then the call to method Check performs calculation of the function based on current data and return 0 if successfull or non 0 if not successfull. Also this property is set to true after this call.

.

Syntax:

```
Property IsChecked As Boolean
```

Level

Description:
If the function is used in some other function then this function will have level 2 while its parent function will have level 1, this goes in depth depending on how many dependant function you have. It is needed when calculating the functions because those functions with higher level need to be calculated first in order that functions with lower level are calculated properly.

Syntax:

```
Property Level As Long
```

Explanation of the parameters:

`FilterID`
`level number`

2.10.4Class IFunction

Methods	Version Availability	Properties	Version Availability
Check	4,5	IsChecked	4,5
		IsUsed	4,5
		Level	4,5

Check

Description:
If the IsChecked property is set to false then the call to this method performs calculation of

the function based on current data and return 0 if successfull or non 0 if not successfull. Also the IsChecked property is set to true after this call.

Syntax:

```
Function Check() As Long
```

IsChecked

Description:
If this property is set to false then the call to method Check performs calculation of the function based on current data and return 0 if successfull or non 0 if not successfull. Also this property is set to true after this call.

.

Syntax:

```
Property IsChecked As Boolean
```

IsUsed

Description:
Property tells you if this function is used on the label in some other function, element, .

Syntax:

Property IsUsed As Boolean

Level

Description:
If the function is used in some other function then this function will have level 2 while its parent function will have level 1, this goes in depth depending on how many dependant function you have. It is needed when calculating the functions because those functions with higher level need to be calculated first in order that functions with lower level are calculated properly.

Syntax:

```
Property Level As Long
```

Explanation of the parameters:

FilterID
level number

2.10.5Class ILabel

Methods	Version Availability	Properties	Version Availability
GetPrinterDarknessList Values	3,4,5	OriginalPrintID	5
GetPrinterList	4,5		
GetPrinterSpeedListVal ues	4,5		

GetPrinterDraknessListValues

Description:
Method retrieves commma separated list of darkness values for specified printer.

Syntax:

```
Function GetPrinterDarknessListValues(PrinterName As String) As
String
```

GetPrinterList

Description:
Method retrieves comma separated list of printer names.

Syntax:

```
Function GetPrintersList() As String
```

GetPrinterSpeedListValues

Description:
Method retrieves commma separated list of speed values for specified printer.

Syntax:

```
Function GetPrinterSpeedListValues(PrinterName As String) As
String
```

OriginalPrintID

Description:
With this property you can set the original print id for the print job before print action. It is usually used for reprint when you want to set the original print id so that you know from which print job this print job was reprinted.

Syntax:

```
OriginalPrintID As String
```

2.10.6Class IParagraph

Methods	Version Availability	Properties	Version Availability
		CharSet	3,4,5
		IsRTL	3,4,5
		ScaleFactor	3,4,5

CharSet

Description:
Property returns character set for this object.

Syntax:

```
Property CharSet As Long
```

IsRTL

Description:
Property returns true if the text is in "right to left" mode.

Syntax:

```
Property IsRTL As Boolean
```

ScaleFactor

Description:
Returns font scaling percentage.

Syntax:

```
Property ScaleFactor As Long
```

2.10.7Class IRTFText

Methods	Version	Properties	Version

	Availability		Availability
		CharSet	3,4,5
		IsRTL	3,4,5

CharSet

Description:
Property returns character set for this object.

Syntax:

```
Property CharSet As Long
```

IsRTL

Description:
Property returns true if the text is in "right to left" mode.

Syntax:

```
Property IsRTL As Boolean
```

2.10.8Class IVar

Methods	Version Availability	Properties	Version Availability
GetFormatChars	4,5	HasPickList	4,5
GetSelectedListElement	4,5		

GetFormatChars

Description:
Method returns allowed characters for this variable.

Syntax:
```
Function GetFormatChars() As String
```

GetSelectedListElement

Description:
If the HasPickList property is true then the variable has a picklist and this method returns the curently selected item in the picklist.

Syntax:

```
Function GetSelectedListElement() As String
```

HasPickList

Description:
Property returns true if the variable is defined as pick list, otherwise returns false.

Syntax:

Property HasPickList As Boolean

2.11 Advanced printing via ActiveX interface

When printing from NiceLabel no information about the actual printout is returned. Now with this new ActiveX calls user can retrieve an exact status of the print job. To use this new functionality you need to know two interfaces and use two methods that were introduced. The first one prints the label and returns the ID for the corresponding print job. It is PrintAndGetJobID and it is described in Class IniceLabel chapter.
The second one GetPrintJob is described below and it is part of the Nlog5 interface

2.11.1 How to use Nlog5 Interface

You must include Nlog5 type library in your project. It is installed together with Nicelabel itself. Deafult location is **C:\Program Files\Common Files\EuroPlus Shared** file Nlog5.dll.
Nlog5.dll and NLog5_TLB.pas are included in the **Nicelabel Integration Documents and Samples** package which is a part Nicelabel 5 distribution CD

The following procedures show how to install and use the type library with Visual Basic, Delphi and Visual Studio.

Visual Basic

You must include Nlog5 in your project:

- Choose Project -> References.
- Activate Nlog5 in the list of available references and validate the dialog box.

Delphi

You must include Nlog5 in your project:

- Choose Project -> Import Type Library
- Activate Nlog5 in the list of available references and validate the dialog box.

.Net – Microsoft Visual Studio

You must include Nlog5 in your project:

- Choose Project -> Add References.
- Activate Nlog5.dll in the list of available references in .COM tab

With PrintAndGetJobID (**refer to Nicelabel Aplicattion Class**) and GetPrintJob (**refer to Nlog5 IPrintJobFactory Class**) calls user can retrieve an exact status of the print job. The first one prints the label and returns the ID for the corresponding print job. The second one returns an interface to a PrintJob object. This object contains a member called Status, detailed status of a print job

2.11.2 Class IPrintJobFactory

Methods	Version Availability	Properties	Version Availability
Init	5		
GetPrintJob	5		

Init

Description:
This method has to be executed before GerPrintJob method

Syntax:

Function Init(OEM As Long, Language As Long) As Long

Parameters:

OEM

`0 – only valid value`

Language

`1 – only valid value`

GetPrintJob

Description:

This method returns an interface to a PrintJob object. This object contains a member called Status. This is an integer field containing the detailed status of a print job. Status is returned as flags, meaning one job can have multiple values for example it can be in state Printing, Error and Deleting at the same time. For more information refer to MSDN Spooler statuses.

Statuses and their meaning:

Status Name	Status Flag	Meaning
StartSpooling	1	Print Job is being sent to Spooler
PartiallySpooled	2	Part of the Print Job is already sent to Spooler
SpoolOk	4	Print Job was sent to Spooler with no errors
SpoolFailed	8	Print Job failed while sending it to Spooler
Deleting	16	Print Job is marked for deletion in Spooler
Deleted	32	Print Job was deleted from Spooler
Error	64	Print Job is in error state in Spooler
Paused	128	Print Job is paused in Spooler
Printed	256	Print Job was sent to Printer
Printing	512	Print Job is being sent to Printer
Restarting	1024	Print Job is restarting
Spooling	2048	Print Job is Spooling
Queued	4096	Print Job is Queued

Syntax:

Function GetPrintJob(ID As String) As PrintJob

2.12 NiceForm ActiveX Interfaces

2.12.1 Introduction

NiceForm has two types of ActiveX interfaces. Each of them has specific purpose, the first interface is meant for design mode and the second is meant for the run mode.

Design mode:

Main purpose of using ActiveX for design mode is to open form in one of the possible ways which offers you different possibilities of designing the form.

For more information refer to the NiceForm ActiveX Interface – Design mode chapter.

Run mode:

The only purpose of the run mode is to run desired form directly from your application. There is also possibility to execute the form also from the design mode interface, but if you just need to run the form without design operation this interface is recommended.

For more information refer to the NiceForm ActiveX Interface – Run mode chapter.

2.12.2 NiceForm ActiveX Interface - Design mode

To use NiceForm ActiveX Interface in design mode use the following class in your application:

Class name

NiceForm5.Designer

Class IApplication

Methods	Version Availability	Properties	Version Availability
Design	3,4,5	FormID	3,4,5
Login	3,4,5		
Run	3,4,5		

Design

Description:

Design method opens NiceForm in design mode. There are two available parameters and therefore four combinations how this method can be used:

1. **FormFileName and LabelFileName are specified.**

In this case “New Form Wizard” in NiceForm is started. Wizard takes the specified label and store the form as the FormFileName parameter.

2. **Only LabelFileName is specifed and FormFileName is empty**

In this case “New Form Wizard” in NiceForm is started. Wizard takes the specified label and store the form as specified manually in the wizard.

3. **Only FormFileName is specified and LabelFileName is empty**

If provided FormFileName exist then this form is opened, otherwise new form is created and FormFileName is set as form file name.

4. **Both FormFileName and LabelFileName are empty**

When you do not specify both FormFileName and LabelFileName then empty form is opened.

Syntax:

```
Function Design(FormFileName As String, LabelFileName As String)
As Boolean
```

Login

Description:

Login method is used, when login is required. As Username you must specify the user, with which you want to open NiceForm.

If Login is successful the TRUE result is returned.

Syntax:

```
Function Login(UserName As String) As Boolean
```

Run

Description:

NiceForm can be started in the run mode. For this you can use Run method. As a parameter you must provide the file name to the form you wish to run.

If form was ran ok then TRUE result is retured.

Syntax:

```
Function Run(FormFileName As String) As Boolean
```

FormID

Description:

FormID property returns the ID of the opened form.

Syntax:

```
Property FormID As Long
```

2.12.3 NiceForm ActiveX Interface - Run mode

To use NiceForm ActiveX Interface in run mode use the following class in your application:

Class Name

NiceForm5.Engine

Class IApplication

Methods	Version Availability	Properties	Version Availability
Login	3,4,5	FormID	3,4,5
Open	3,4,5		

Open

Description:

To run a form in run mode use the Open method. As a parameter you must provide the file name to the form you wish to run.

If form was ran ok then TRUE result is retured.

Syntax:

```
Function Open(FormFileName As String) As Boolean
```

Login

Description:

Login method is used, when login is required. As Username you must specify the user, with which you want to open NiceForm.

If Login is successful the TRUE result is returned.

Syntax:

```
Function Login(UserName As String) As Boolean
```

FormID

Description:

FormID property returns the ID of the executed form.

Syntax:

```
Property FormID As Long
```

3. LabelServices Interface

3.1 Introduction

LabelServices interface allows a 3rd party application to supervise printer statuses.

A service called Label Services (LblServices.EXE) is installed and started on each client after the Nicelabel Pro installation. Label Services also runs a sub-process NDH.EXE. Both services are monitoring the local printer and its print job statuses and can report the statuses back to the labeling application where the statues are continuously updated in the log file.

For NiceLabel SDK Users:
NiceLabel SDK now includes the option of a Printer Status Monitoring Service. This adds NiceLabel Services (with externals) to the system.

If you stop the Label Service, the functionality will not be available.

To access print queue soap interface application should connect to http://localhost:6758/wsdl/PrintQueueSOAP

3.2 Class TPQPrinter

Properties	Version Availability
Attributes	5
DriverName	5
HasWebBrowser	5
Name	5
NiceStatus	5
PrintJobList	5
ShareName	5
Status	5
Updated	5
WebAddress	5

Attributes

Description:

Specifies the printer attributes.

Type:

`Integer`

Possible values:

`PQ_PRINTER_ATTRIBUTE_NETWORK = $00000001`
`Printer is a network printer connection.`

`PQ_PRINTER_ATTRIBUTE_NICE =    $00000002`
`Printer was installed using nice drivers.`

`PQ_PRINTER_ATTRIBUTE_SHARED =  $00000004`
`Printer is shared.`

`PQ_PRINTER_ATTRIBUTE_OFFLINE = $00000008`
`Indicates whether the printer is currently connected.`

`PQ_PRINTER_ATTRIBUTE_DEFAULT = $00000020`
`Indicates that the printer is the default printer on the system.`

DriverName

Description:

Printer driver name.

Type:

`String`

HasWebBrowser

Description:

True - printer has built in web server.
False - printer has not built in web server.

Type:

`Boolean`

Name

Description:

Specifies the printer name.

Type:

`String`

NiceStatus

Description:

List of string representing advanced errors on printers such as head open, out of ribbon,….

Type:

`TPQNiceStatus`

<u>PrintJobList</u>

Description:

List of the currently queued print jobs.

Type:

`TPQPrintJobList`

<u>ShareName</u>

Description:

Identifies share name of the printer. Only used if `PQ_PRINTER_ATTRIBUTE_SHARED` is set for attributes member.

Type:

`String`

<u>Status</u>

Description:

Specifies the printer status.

Type:

`Integer`

Possible values:

`PQ_PRINTER_STATUS_PAUSED = $00000001`
`Printer is paused.`

`PQ_PRINTER_STATUS_PRINTING = $00000002`
`Printer is currently printed.`

`PQ_PRINTER_STATUS_ERROR = $00000004`
`Printer is in error state.`

`PQ_PRINTER_STATUS_NOT_AVAILABLE = $00000008`
`The printer is not available for printing.`

`PQ_PRINTER_ATTRIBUTE_DEFAULT = $00000020`
`Indicates that the printer is the default printer on the system.`

<u>Updated</u>

Description:

True - any of the members were updated.
False – no members were updated

Type:

`Boolean`

3.3 Class TPQPrinterList

Properties	Version Availability
`PrinterArray`	5

<u>PrinterArray</u>

Description:

List of TPQPrinter classes.

Type:

`TPQPrinterArray`

3.4 Class TPQPrinterJob

Properties	Version Availability
OEM	5
ID	5
Name	5
Pages	5
PagesPrinted	5
Position	5
PrinterName	5
Priority	5
Quantity	5
Status	5
Owner	5
Submitted	5

OEM

Description:

Has to be 0.

Type:

`Integer`

ID

Description:

Specifies a print job identifier.

Type:

`Integer`

Name

Description:

Specifies the name of print job.

Type:

`String`

Pages

Description:

Specifes number of pages the print job contains.

Type:

`Integer`

PagesPrinted

Description:

Specifies the number of pages that have printed.

Type:

`Integer`

Position

Description:

Specifies the print job’s position in the print queue.

Type:

`Integer`

PrinterName

Description:

Specifies the name of the printer for which the print job was spooled.

Type:

`String`

Priority

Description:

Specifies the priority of the print job.

Type:

`TPQPrintJobPriority`

Quantity

Description:

Specifies the number of labels that will be printed.

Type:

`Integer`

Status

Description:

Specifies the print job status.

Type:

`Integer`

Possible values:

```
PRINTJOB_STATUS_PARTIALLY_SPOOLED = $00000002
User cancelled print job aused.
PRINTJOB_STATUS_SPOOL_OK =  $00000004
Print job was successfully spooled.
```

PRINTJOB_STATUS_SPOOL_FAILED = $00000008
There was an error in label production such as wrong barcode value.

PRINTJOB_STATUS_DELETING = $00000010
Print job is being deleting.

PRINTJOB_STATUS_DELETED = $00000020
Print job has been deleted.

PRINTJOB_STATUS_ERROR = $00000040
An error is associated with print job.

PRINTJOB_STATUS_PAUSED = $00000080
Print job is paused.

PRINTJOB_STATUS_PRINTING = $00000100
Print job is printing.

PRINTJOB_STATUS_PRINTED = $00000200
Print job is printed.

PRINTJOB_STATUS_RESTART = $00000400
Print job has been restarted.

PRINTJOB_STATUS_SPOOLING = $00000800
Print job is spooling.

PRINTJOB_STATUS_QUEUED = $00001000
Print job is queued.

PRINTJOB_STATUS_MOVED = $00002000
Print job has been moved to another printer.

PRINTJOB_STATUS_PREPARING = $00004000
Language monitor is preparing job to be sent to printer.

PRINTJOB_STATUS_SENDING = $00008000
Language monitor is sending the print job to printer.

PRINTJOB_STATUS_SENT_OK = $00010000
Language monitor successfully sent the print job to the printer.

PRINTJOB_STATUS_SEND_FAILED = $00020000
Language monitor failed at sending print job to printer.

Owner

Description:

Specifies the name of the user that owns the print job.

Type:

String

Submitted

Description:

Specifies the time that this print job was spooled.

Type:

`TXSDateTime`

3.5 Class TPQPrinterJobList

Properties	Version Availability
`PrinterArray`	5

PrintJobArray

Description:

List of TPQPrintJob classes.

Type:

`TPQPrintJobArray`

3.6 Types

Properties	Version Availability
`TPQPrintJobPriority`	5
`TPQPrintJobMoveDirection`	5
`TPQPrinterArray`	5
`TPQNiceStatus`	5
`TPQPrintJobArray`	5
`TPrinterNames`	5
`TPrintJobIds`	5

TPQPrintJobPriority

Description:

`Print job priority.`

Type:

Enum:(pjpHigh, pjpNormal, pjpLow)

TPQPrintJobMoveDirection

Description:

Print job move direction.

Type:

Enum:(pjmvup, pjmvDown)

TPQPrinterArray

Description:

List of TPQPrinter classes.

Type:

Array of TPQPrinter

TPQNiceStatus

Description:

List of advanced printer.

Type:

Array of String

TPQPrintJobArray

Description:

List of TPQPrintJob classes.

Type:

Array of TPQPrintJob

TPrinterNames

Description:

List of printer names.

Type:

Array of String

TPrintJobIds

Description:

List of print job identifiers.

Type:

Array of Integer

3.7 Methods

Methods	Version Availability
ChangePrintJobPosition	5
ChangePrintJobPriority	5
DeleteAllPrintJobs	5
DeletePrintJobs	5
GetPrinters	5
GetPrintersInErrorState	5
GetPrinterInfo	5
GetPrintJobs	5
GetPrinterNames	5
GetPrinterJobsCount	5
GetSpecifiiedPrinters	5
GetSpecifiiedPrintersInErrorState	5
MovePrintJobs	5
PausePrintJobs	5
PausePrinters	5
ProcessJob	5
RenamePrinter	5
RestartPrintJobs	5
SharePrinter	5
UsePrintersOffline	5
UsePrintersOnline	5

ChangePrintJobPosition

Description:
Moves print jobs one position up or down in queue list.

Syntax:

```
procedure ChangePrintJobsPosition(const APrintJobIDs:
TPrintJobIds; AMoveDirection: TPQPrintJobMoveDirection)
```

Parameters:

APrintJobIDs
Print job ids list. Only the print jobs specified in this list will be used.

AMoveDirection
Set this to pjmvUp to move it one position up or set it to pjmvDown to move it one position down

ChangePrintJobPriority

Description:
Change the priority of specified print jobs. Print job with higher priority will be printed before the ones with lower priority no matter the position in the queue.

Syntax:

```
procedure ChangePrintJobsPriority(const APrintJobIDs:
TPrintJobIds; APriority: TPQPrintJobPriority)
```

Parameters:

APrintJobIDs
Print job ids list. Only the print jobs specified in this list will be used.

APriority
Possible values are: pjpHigh, pjpNormal, pjpLow. Default priority is pjpLow.

DeleteAllPrintJobs

Description:
Delete all print jobs currently in queue for the specified printers.

Syntax:

```
procedure DeleteAllPrintJobs(const APrinterNames: TPrinterNames)
```

Parameters:

APrinterNames

```
Printer names list. Only the printers specified in this list will be
used.
```

DeletePrintJobs

Description:
Deletes specified print jobs.

Syntax:

```
procedure DeletePrintJobs(const APrintJobIDs: TPrintJobIds)
```

Parameters:

`APrintJobIDs`

```
Print job ids list. Only the print jobs specified in this list will be
used
```

GetPrinters

Description:
Retrieves all installed printers. Note that this function does not retrieve the printers that are connected from other machines, it retrieves only locally installed printers.

Syntax:

```
function GetPrinters(const ALastUpdate: TDateTime; var AAllPrinterNames:
WideString; const ANicePrintersOnly: Boolean): TPQPrinterList)
```

Parameters:

`ALastUpdate`

```
If not 0 then it retrieves only the printers that were modified after the
time specified in ALastUpdate. For example if ALastUpdate is 12:00 and
printer was modified at 11:55 then this printer will not be added to the
printer list.
```

`AAllPrinterNames`

```
Returns names of all installed printers separated by comma.
```

`ANicePrintersOnly`

Set this to TRUE to specify that only the printers that were installed using nice drivers should be returned .

GetPrintersInErrorState

Description:
Returns printers that meet the following conditions:
- printer's status is ERROR
- printer have job(s) in queue and it is paused or offline

Syntax:

```
function GetPrintersInErrorState(): TPQPrinterList
```

GetPrinterInfo

Description:

Returns printers that meet the following conditions:

- printer's status is ERROR
- printer have job(s) in queue and it is paused or offline

Syntax:

```
function GetPrinterInfo(const APrinterName: WideString; out
ASpoolerStatus: Integer; out APrinterStatus: TPQNiceStatus): Boolean
```

Parameters:

APrinterName
Name of printer for which info should be retrieved..

ASpoolerStatus
Current status of the printer as seen in the spooler.

APrinterStatus
List of string representing advanced printer errors.

GetPrintJobs

Description:

Returns print jobs for specified printers.

Syntax:

```
function GetPrintJobs(const ALastUpdate: TDateTime; const APrinterNames:
TPrinterNames; var AAllPrintJobIDs: WideString): TPQPrintJobList
```

Parameters:

ALastUpdate
If not 0 then it returns only the print jobs that were modified after the time specified.

APrinterNames
Comma separated string of printer names for which print jobs should be returned.

AAllPrintJobIDs
Return ids of all print jobs for specified printers, separated by comma.

GetPrinterNames

Description:
Retreives all installed printer names .

Syntax:

```
function GetPrinterNames():WideString
```

GetPrinJobsCount

Description:
Returns count of all print jobs for all printers.

Syntax:

```
function GetPrinterJobsCount(): WideString
```

GetSpecifiedPrinters

Description:
Returns count of all print jobs for specified printers.

Syntax:

```
function GetSpecifiedPrinters(var APrinterNames: WideString):
TPQPrinterList
```

Parameters:

APrinterNames
Comma separated string of printer names for which print jobs should be returned.

GetSpecifiedPrintersErrorState

Description:
Returns for specified printers that meet the following conditions:
- printer's status is ERROR
- printer have job(s) in queue and it is paused or offline

Syntax:

```
function GetSpecifiedPrintersInErrorState(var APrinterNames: WideString):
TPQPrinterList
```

Parameters:

APrinterNames
Comma separated string of printer names for which print jobs should be returned.

MovePrintJobs

Description:
Moves print jobs to another printer.

Syntax:

```
function MovePrintJobs(const APrintJobIDs: TPrintJobIds; APrinterName:
WideString): Boolean
```

Parameters:

APrinteJobIDs
Print job ids list. Only the print jobs specified in this list will be used.

APrinterName
Name of printer to which print jobs should be moved.

MovePrintJobsEx

Description:
Moves print jobs to another printer and returns list of generated errors in TMoveErrors – array of TMoveError(class TRemoteable)

Syntax:

```
function MovePrintJobsEx(const APrintJobIDs: TPrintJobIds; APrinterName:
WideString): TMoveErrors
```

Parameters:

APrinteJobIDs
Print job ids list. Only the print jobs specified in this list will be used.

APrinterName
Name of printer to which print jobs should be moved.

PausePrintJobs

Description:
Pauses / resumes specified print jobs.

Syntax:

```
procedure PausePrintJobs(const APrintJobIDs: TPrintJobIds; APause:
Boolean)
```

Parameters:

APrinteJobIDs
Print job ids list. Only the print jobs specified in this list will be used.

APause
If TRUE print jobs will be paused else they will be resumed.

PausePrinters

Description:
Pauses / resumes specified printers.

Syntax:

```
procedure PausePrinters(const APrinterNames: TPrinterNames; APause:
Boolean)
```

Parameters:

APrinterNames
Printer names list. Only the printers specified in this list will be used.

APause
If TRUE printers will be paused else they will be resumed.

ProcessJob

Description:
Generates JOB File for specified OEM

Syntax:

```
function ProcessJob(const AOEM: Integer; const AJobContents: WideString):
Boolean
```

Parameters:

AOEM
OEM value, use 0 for NiceLabel.

AJobContents
JOB file contents.

RenamePrinter

Description:
Renames specified printer.

Syntax:

```
procedure RenamePrinter(const APrinterName: WideString; ANewPrinterName:
WideString)
```

Parameters:

APrinterName
Existing printer name.

ANewPrinterName
New printer name

RestartPrintJobs

Description:
Restarts print jobs. This means that the spooler will resend whole job again.

Syntax:

```
procedure RestartPrintJobs (AprintJobIDs: array of Integer)
```

Parameters:

APrintJObIDs
Print job ids list. Only the print jobs specified in this list will be used.

SharePrinter

Description:
Sets/Removes share mode for the specified printer.

Syntax:

```
procedure SharePrinter(const APrinterName: WideString; const AShare:
Boolean; const AShareName: WideString)
```

Parameters:

APrinterName
Name of the printer that will be shared.

AShare
If TRUE printer will be shared, else shared flag will be removed.

AShareName
Name displayed to the remote computers that will connect to this printer.

UsePrintersOfflLine

Description:
Sets OffLine mode for specified printers.

Syntax:

```
procedure UsePrintersOffline(const APrinterNames: TPrinterNames)
```

Parameters:

APrinterName
Name of the printer that will be shared.

UsePrintersOnLine

Description:
Sets OnLine mode for specified printers.

Syntax:

```
procedure UsePrintersOnline(const APrinterNames: TPrinterNames)
```

Parameters:

APrinterName
Name of the printer that will be shared.

4. DDE Communication

4.1 Introduction

To create the DDE communication, the client application must use the following DDE parameters:

```
Service = NiceLabel4
```

```
Topic = LINE or JOB
```

When you are using the JOB topic, the content is the name of the command file that must be run. When you are using the LINE topic, the content is one of the NiceLabel commands.

When you want to use DDE communication to control NiceLabel, it is probably best, when the user doesn't know for the background running of this application. For this purpose you can use this command parameter:

```
NICE5.EXE <label_name> /s
```

The parameter s (silent) prevents NiceLabel from being visible on the screen. It will run in a minimized form.

4.2 DDE Commands

4.2.1 DDEInitiate

Description:

DDEInitiate command initiates a link between NiceLabel and your application.

Syntax:

```
DDEInitiate(Service, Topic)
```

4.2.2 DDEExecute

Description:

DDEExecute command executes a command on your label. For list of available commands please see NiceCommands chapter of this manual.

Syntax:

```
DDEExecute Channel Command
```

4.2.3 DDETerminate

Description:

DDETerminate command terminates a link between NiceLabel and your application.

Syntax:

```
DDETerminate Channel
```

4.3 NiceCommands

Printing with NiceLabel can accomplished automatically. There are several methods for automation. The first method is with the use of command files (JOB file), which is used with the **Automatic print** command from the **File** menu or directly from the shell.

Another way is with the help of any other Windows application, which enables DDE communication between the applications.

In both methods, the same commands may be used. When you are using the automatic print, the commands must be written one per line in the command file (JOB file). With DDE communication, the commands are sent through the DDE channel.

All NiceCommands are presented in four main groups:

- COMMON

  NiceCommands which are used for common use.
- FILE

  NiceCommands which are used for file operations.
- PRINTING

  NiceCommands which are used for printing operations.
- DATABASE

  NiceCommands which are used for database operations.

4.3.1 COMMON

SET

Description:

Name is the name of the variable defined on the label. If the variable isn't on the label, an error will occur. `Step` and `Quantity_of_repetition` are optional parameters. These parameters specify the increment of the variable and the number of the labels before change.

If variable value contains space characters or commas, you have to enclose the text in quotation marks.

Syntax:

```
SET name = value_of_the_variable, [,step[,
quantity_of_repetition]]
```

COMMENT

Description:

When developing program code or scripts it is very wise to well document your commands. This will help you decode what the script really performs, when you will look at the code after some time.

Use semicolon (;) on the beginning of the line. Everything following it will be treated as script comment and will not be processed by NiceLabel.

Syntax:

```
;
```

LOGIN

Description:

Performs login procedure into NiceLabel program. This is necessary when login into NiceLabel is required.

NOTE. This is DDE command and cannot be used in .JOB files.

Syntax:

```
LOGIN <username>
```

RETURN

Description:

This command returns to the NiceLabel program after printing.

Syntax:

```
Return
```

QUIT

Description:

This command stops the NiceLabel program after printing. The application is closed.

Syntax:

```
Quit
```

MESSAGEBOX

Description:

Print the message. The second parameter represents the title of the message dialog box.

If variable value contains space characters or commas, you have to enclose the text in quotation marks (e.g. `MESSAGEBOX "Insert labels in printer", Warning`).

Syntax:

```
MESSAGEBOX message [, caption]
```

OEMTOANSI

Description:

This command works in conjunction with command `SET`. It puts the text that follows the command `SET` in proper codepage, so that variable is assigned the proper value.

Use it to put the values following SET command to the proper codepage, so correct values will be transferred to NiceLabel at print time.

Syntax:

```
OEMTOANSI ON|OFF
```

IGNOREERROR

Description:

This command enables/disables ignoring of an error, that occurs during command file processing. If ignored, the command file will continue to be executed after wrong value for variable is set for example.

Syntax:

```
IGNOREERROR ON|OFF
```

TEXTQUALIFIER

Description:

This command sets the data delimiter for SET command to the specified character.

Syntax:

```
TEXTQUALIFIER <<character>>
```

4.3.2 FILE

CREATEFILE

Description:

This command lets you create the file with the contents NiceLabel.

The purpose of creating or deleting files is that the client application knows when will be printing stopped.

The example of usage is printing from the file. First the application prepares variable data for the labels into particular file. Now NiceLabel is activated and printing starts. To inform application, that the printing process is finished, file with data could be deleted at the end. This could be a signal to application, that new "job" could be started.

See also:

DELETEFILE

Syntax:

```
CREATEFILE <name_of_the_file>
```

DELETEFILE

Description:

This command deletes the file.

See also:

CREATEFILE

Syntax:

DELETEFILE <name_of_the_file>

LABEL

Description:

The LABEL command opens the working label. If the label is already opened, the program will use this instance. It is recommended to write the full path name along with the file name. When printername parameter is provided, label is opened for that printer.

Note, if the variable value contains space characters or commas, you will need to enclose the whole path in quotation marks (e.g. `LABEL "C:\Program Files\EuroPlus\Labels\sample3.lbl"`).

If you use the LABEL command with NiceWatch running in service mode, use the UNC quotation instead of the mapped drives (e.g. LABEL `"\\SERVER\MY LABELS\LABEL.LBL"` instead of `"G:\MY LABELS\LABEL.LBL"`).

Syntax:

```
LABEL <name_of_the_file,[printername] >
```

FILECLOSE

Description:

The FILECLOSE command closes the currently active label. NiceLabel will stay opened.

Syntax:

```
FILECLOSE
```

LABELCLOSE

Description:

The LABELCLOSE command closes the currently active label. NiceLabel will stay opened.

This command is introduced as a synonym for FILECLOSE command.

Syntax:

```
LABELCLOSE
```

EXPORTLABEL

Description:

The EXPORTLABEL command exports currently active label. NiceLabel will stay opened.

Syntax:

EXPORTLABEL

4.3.3 PRINTING

PRINT

Description:

The `Print` command starts the form or label printing. The first parameter is the print quantity of the labels; it can be a number, or one of following words:

- VARIABLE
- UNLIMITED

The first parameter means printing on the base of the variable quantity (one of the variables sets the quantity), the second one means unlimited printing (printing from the whole database file for example).

Second parameter in the command represents the number of the labels you want to omit before first printed label on the page. The parameter can be used when the part of the page is already printed. The rest of the unused labels on the page can be printed with the help of this parameter.

Syntax:

```
PRINT quantity[, skip]
```

<u>PORT</u>

Description:

This command overrides the printer's port name. Next `PRINT` command will print to the port specified.

Usually this command is used to print the label to a file. In this case you must specify name of file in parameter `port_name` before using the PRINT command.

Syntax:

```
PORT <port_name>
```

<u>PRINTER</u>

Description:

Normally the `PRINT` command prints the label using the printer specified in the label file. With this command you can override the printer and print the labels using different printer.

If printer name contains space characters, you have to enclose it in quotation marks.

For `printer_name` always use the system printer name as is displayed in the status line in the NiceLabel Pro application. System printer names are usually the same as the printer names in Printers folder from Control Panel. They differ only when you are using network-connected printers, when you should use "\\server\share" syntax and not a printer friendly name.

Syntax:

```
PRINTER <printer_name>
```

<u>PRINTJOBNAME</u>

Description:

Specifies the print job name that will be used in print manager when using `PRINT` command. After printing the name is returned in normal state.

If variable value contains space characters or commas, you have to enclose the text in quotation marks (e.g. `PRINTJOBNAME "Label for printing"`).

Syntax:

```
PRINTJOBNAME <job_name>
```

SESSIONSTART

Description:

All three commands `(SessionStart, SessionPrint, SessionEnd)` are used together. If ordinary command `SessionPrint` is used, every time a complete data stream for printer is sent. If you want to join multiple Print commands into one data stream, you can use the command `SessionStart` followed with any number of `SessionPrint` commands and in the end use the command `SessionEnd`. The stream is not closed until the command `SessionEnd` occurs. These commands offer a way of optimal printing through NiceCommands and it is not necessary to generate a complete data stream for each print session.

See also:

SESSIONPRINT

SESSIONEND

SESSIONPRINT

Description:

```
SESSIONPRINT quantity [, skip]
```

You send the data stream to printer using this function. You can use multiple `SessionPrint` commands one after another and join them in single data stream. The stream is not closed until the command `SessionEnd` occurs. The meaning of quantity and skip parameters is the same as with Nice Command `PRINT`.

See also:

PRINT

SESSIONSTART

SESSIONEND

SESSIONEND

Description:

The function closes data stream.

See also:

SESSIONSTART

`SESSIONPRINT`

SETPRINTPARAM

Description:

This command allows you to set advanced print parameters before printing.

Currently supported PARAMNAME are:

PAPERBIN: Use it to specify from which tray the paper should be used.

If the printer is equipped with more than just one paper / label tray, you can control which is used for printing. The name of the tray should be acquired from the printer driver.

PRINTSPEED: Use this parameter to specify printing speed. The value for parameter varies from one printer to the other. Consult printer's manuals for numbers.

PRINTDARKNESS: Use this parameter to specify printing darkness / contrast. The value for parameter varies from one printer to the other. Consult printer's manuals for numbers.

PRINTOFFSETX: Use this parameter to specify left offset for all printing objects. The value for parameter must be numeric, positive or negative, in pixels

PRINTOFFSETY: Use this parameter to specify top offset for all printing objects. The value for parameter must be numeric, positive or negative, in pixels

Syntax:

```
SETPRINTPARAM <paramname> = <value>
```

4.3.4 DATABASE

SETDATABASE

Description:

Value	Description
database_name	The name of the currently used database as defined in the program
value	The name of the new table that should be used as data source

This command allows you to use some other database with the label file and not the one that was connected to the label file at design time.

This other database will only be used when printing labels, the label file will remain intact with connection to the original database.

See also:

SETTABLE

Syntax:

```
SETDATABASE <database_name> = <value>
```

SETTABLE

Description:

Value	Description
table_name	The name of the currently used table as defined in the program
value	The name of the new table that should be used as data source

This command allows you to use some other table with the label file and not the one that was connected to the label file at design time.

This other database table will only be used when printing labels, the label file will remain intact with connection to the original table.

The new database table should be of the same type as original table. For example, you cannot change the table from dBase to Paradox. The structure of new table has to be identical to the original one.

You can use table from the database that is already connected to the label or from some entirely different database.

See also:

SETDATABASE

Syntax:

```
SETTABLE <table_name> = <value>
```

5. Programming Samples

5.1 DDE

DDE programming samples for Microsoft Excel, Microsoft Word are included in the **Nicelabel Integration Documents and Samples** package which is a part Nicelabel 5 distribution CD.

5.2 ActiveX interface

ActiveX programming samples for Visual Basic, Visual Studio and Borland Delphi are included in the **Nicelabel Integration Documents and Samples** package which is a part Nicelabel 5 distribution CD.

5.3 ActiveX Hints

Here is described how to speed up printing with NiceLabel through ActiveX interface. Following you will find four options how communication can be performed. They are sorted from the slowest to the fastest one.

<u>VERY VERY SLOW:</u>

Description:

The main reason this option is the slowest one is that every time when label is going to be opened the new NiceLabel object is created. After that every label is opened and closed for every printout. At the end NiceLabel software is closed as well.

Syntax:

```
createObject (NiceLabel5)
LabelID = niceApp.LabelOpen("sample3.lbl")
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)
nice.Quit

createObject (NiceLabel5)
LabelID = niceApp.LabelOpen("sample3.lbl")
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)
nice.Quit

createObject (NiceLabel5)
LabelID = niceApp.LabelOpen("sample3.lbl")
```

```
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "33", 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)
nice.Quit
```

SLOW:

Description:

This is a faster option than the previous one but it is still slow. Now the object NiceLabel is created just once for the whole print process. This speeds us the response times and labels are printed faster. But every time the label is printed, it is also immediately closed. When the next print of the same label is required, it must be re-opened. The operation takes some time.

Syntax:

```
createObject (NiceLabel5)
LabelID = niceApp.LabelOpen("sample3.lbl")
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)

LabelID = niceApp.LabelOpen("sample3.lbl")
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)

LabelID = niceApp.LabelOpen("sample3.lbl")
 Success = niceApp.LabelSetVar(LabelID, "Bar Code", "33",
 0, 0)
 Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelClose(LabelID)
nice.Quit
```

FAST:

Description:

This is one of the fastest options. NiceLabel software is started only once. The label is also opened only once and then just left opened in the NiceLabel software. When additional label printout is required, the already opened label will be used again.

Syntax:

```
LabelID = niceApp.LabelOpen("sample3.lbl")

Success = niceApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label
Success = niceApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label

Success = niceApp.LabelSetVar(LabelID, "Bar Code", "33", 0, 0)
Success = niceApp.LabelPrint(LabelID, 1) ' Print 1 label

Success = niceApp.LabelClose(LabelID)
```

FASTEST:

Description:

In this case the label opening and closing procedures are performed only once. Additionally the LabelSessionPrint command is used. The main reason is the advantage in speed compared to the common LabelPrint command. The printing actions are defined within a LabelSessionStart and LabelSessionEnd commands. Any number of LabelSessionPrint commands can be used. The labels are cached in the computer memory and they are processed and printed at the same time, when the LabelSessionEnd command is encountered. There is no delay in the label printing as all labels are sent in one batch job. SessionPrint works for one label at a time, so you cannot change the printed label during on print session.

See also:

LabelSessionStart

Syntax:

```
LabelID = niceApp.LabelOpen("sample3.lbl")

niceApp. LabelSessionStart(LabelID)

Success = niceApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
Success = niceApp.LabelSessionPrint(LabelID, 1) ' Print 1 label

Success = niceApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
Success = niceApp.LabelSessionPrint(LabelID, 1) ' Print 1 label

Success = niceApp.LabelSetVar(LabelID, "Bar Code", "33", 0, 0)
Success = niceApp.LabelSessionPrint(LabelID, 1) ' Print 1 label

niceApp. LabelSessionEnd(LabelID)

Success = niceApp.LabelClose(LabelID)
```

5.4 Active Server Pages

ActiveX interface of NiceLabel software gives you the opportunity to add label print functionality to Active Server Pages (ASP) on the internet. Therefore you can achieve true internet label printing capability in your internet solutions. Your internet application can accept the input values from the user, process the data, connect to NiceLabel running in the background and print the labels to some printer, either connected locally or somewhere in the network.

Requirements

To setup an ASP page that you will use to print labels you need to do the following:

- Design a label file
- Create an ASP file

Label file

You have to prepare a label file that will be printed from ASP page. You can use NiceLabel Pro label editor for this task. The labels can be fixed or with variable values. If you have variable object on the label, they are always linked to some variables.

The variables will need to be filled with some values before the printing can occur. Here is where your ASP file comes in. You need to know the names of the variables that are defined on the label so you can set their values. In the sample ASP script we use the variables Name, SurName, Address and Phone.

ASP file

ASP file can be created in any editor that is intended for such operations. ASP file is a text file with the programming commands that ASP web server can understand and execute. ASP file is actually a HTML formatted file enriched with the ASP capabilities. The web page using ASP is no longer static page, but is an output of the ASP program.

When writing the ASP code, you are using the same ActiveX interface to connect to NiceLabel software as you would use when programming the application in Visual Basic, Delphi, C++ and similar development environments.

Sample of the source code can be found in the next paragraph.

Sample ASP file

Here is a sample source code for the ASP web page.

It connects to the label file NiceLabel.lbl, sets the values for the variables as the user has entered and prints one copy of the label to the default printer.

```
<%@ Language = VBScript %>
<% Option Explicit %>

<HTML>
 <HEAD>
 <TITLE>NiceLabel using ASP</TITLE>
 </HEAD>

 <BODY BGCOLOR="White" TOPMARGIN="10" LEFTMARGIN="10">
<FONT SIZE="4" FACE="ARIAL, HELVETICA"> </FONT><img src="NiceLabel.jpg" width="199"
height="65"><BR>
<P>&nbsp;
<FORM NAME=Form1 METHOD=Post ACTION="NiceLabel.asp">
 First Name..........
 <INPUT TYPE=Text NAME=fname>
 <P> Last Name..........
 <INPUT TYPE=Text NAME=lname>
 <P> Address Name....
<INPUT TYPE=Text NAME=address>
 <P> Phone.................
<INPUT TYPE=Text NAME=phone>
 <P>
 <INPUT TYPE=Submit VALUE="Print">

 </FORM>

 <HR>


<%
Dim ID, Result, Nice

Set Nice = CreateObject("NiceLabel5.Application")

ID = Nice.LabelOpen("G:\tmp\ASP\NiceLabel.lbl")
Result = Nice.LabelSetVar(ID, "Name", Request.form("fname"), -9999, -9999)
Result = Nice.LabelSetVar(ID, "SurName", Request.form("lname"), -9999, -9999)
Result = Nice.LabelSetVar(ID, "Address", Request.form("address"), -9999, -9999)
Result = Nice.LabelSetVar(ID, "Phone", Request.form("phone"), -9999, -9999)

Result = Nice.LabelPrint(ID, 1)


Nice.Quit
%>


 </BODY>
</HTML>
```

Complete ASP sample files are installed together with the installation of NiceLabel software. The location for the ASP sample is the folder "..\EuroPlus\NiceLabel 5\Samples\Integration".

Permissions

To successfully use NiceLabel integrated to ASP web page you must correctly set the access permissions on the web server. To set or check the web server permissions run the utility "DCOMCNFG" on the computer running the web server.

1. Expand the Component Services, Computers, My Computer, DCOM Config.
2. Locate NiceLabel in the list of applications with DCOM interface.
3. Right-click it then open its properties.
4. Go to Identity tab and select the option "The interactive user."
5. Go to Security tab ane select the option "Customize" for Launch and Activation Persmissions. Click the button Edit.
6. Select the users that are allowed to run NiceLabel software from the ASP website. These are not external users connecting to the web server, but Windows system users that can run NiceLabel software.

Printing

Label printing through ASP-enabled web server can be performed on all printer drivers that are installed on the server computer. By selecting different ports in the printer drivers, you can redirect printing to different locations.

Those printer drivers can be redirected to the following locations.

- Printers attached directly to the server computer
- Printers attached to the client computer

  If you want to print labels to the client computer you have to redirect printer driver on the server to the client printer port. The printer driver must be installed on the server and you have to make sure that the port the printer is using points to the client computer. Usually the client shares its printer port as (\\server\share), where "server" is the name of the client computer and "share" is the shared name of the port on the client.

- Printers attached to the network

  The port in the printer driver must be redirected to the printer's IP address. In this case the printer has its own network interface card with the IP number and the port number. You need to be familiar with both numbers to correctly set up the network port properties in the printer driver.

5.5 ASP.NET

NiceLabel can be used also in the ASP.NET environment almost on the same way as regular ASP. Basically the only difference is that the local ASP user, which is created when ASP.NET environment is installed, has sufficient permissions to run NiceLabel.

To achieve this you have to set the permissions for that user.

5.6 Visual Basic Script (VBScript)

NiceLabel ActiveX interface can be used also in the Visual Basic (VBScript) environment.
Visual Basic function allows you to perform the most demanding and difficult data manipulation on the label. Available are all the functions, procedures and operands from Microsoft Visual Basic script.

Classes **Application** and **Label** are accessible within VBScript function. All methods and properties of both classes are ready to use in VBScript function. Label class gives you opportunity to use more methods and properties of other classes.

Sample VBSCript

Here is a sample script.
VBscript function below sets “Prompt” variable value to 100.

Method Variables of class Label is used to access VarableList class. With method FindByname class Var is accessed, variable “Prompt” located and SetValue method sets value 100 to variable Prompt

```
a = 100
Result = label.variables.findbyname("Prompt").setvalue(a)
```

6. FAQ

6.1 How to choose different printer for label printing not the one than is selected in the label?

There are two ways how to open the label and work with the NiceLabel ActiveX interface.

If you decide to use **Basic** method then you will have a label ID and you can set printer with LabelSetPrinter command.

```
Status = oNice.LabelSetPrinter(LabelID, "SATO CL408e")
```

The other method is to open the label in **Advanced** mode. In this case you have opened an interface to the label and all label properties are stored in one object. You have to change the PrinterName property to the desired printer name.

```
Status = oLabel.PrinterName = “SATO CL408e”
```

6.2 How to close the label, if you do not have Label ID?

If you do not have the Label ID then you have opened the label in the Advanced mode. You do not have the label ID but you have the interface the label. You have to use the proper command for label close as is available in the Advanced method.

You have to close the application with the Quit method and set the variable to nothing.

```
oNice.Quit

Set oNice = Nothing
```

6.3 How to print label, if you do not have Label ID?

If you do not have the Label ID then you have opened the label in the Advanced mode. You do not have the label ID but you have the interface the label. You have to use the proper command for label print as is available in the Advanced method.

The object you have created for storing of the label interface contains the method Print. Execute the method Print and the label will be printed.

```
Status = oLabel.Print(“1”)
```

6.4 How to save the label with NiceLabel API?

NiceLabel ActiveX interface gives you the print-only functionality of the NiceLabel Pro software. The main purpose of the interface is to provide you with the label printing power in your application. You will print already defined labels.

NiceLabel API does not provide you with label design and save functionality. Labels must be designed in advance in NiceLabel label design environment.

6.5 How can I put NiceLabel in the foreground when I control it with ActiveX interface?

When you use NiceLabel through ActiveX interface it is always running in the background invisible to the user. But you can see it in the list of processes running on the system.

But it is also possible to put NiceLabel in the foreground and then control it with ActiveX interface. Run NiceLabel using the function CreateProcess. The function is part of Windows API. The function returns a handle to NiceLabel print engine that is needed later when you want to close the application.

When NiceLabel is started with the function CreateProcess, call the function GetObject and it will return the interface to the existing instance of NiceLabel.

6.6 How can I use NiceLabel API in .NET environment?

NiceLabel API can be used also in .NET environment. NiceLabel ActiveX on its own does not cooperate with .NET applications quite well. That is why the additional component NiceLabel Wrapper is available to smooth the communication between your .NET application on one end and NiceLabel Pro on the other end.

Whenever the integration to .NET environment is necessary, use the NiceLabel Wrapper.

For more information about NiceLabel Wrapper refer to the chapter **NiceLabel Wrapper** on page 2—14.

6.7 Where can I get programming samples for different environments?

ActiveX samples (label files and source code of the applications) for all programming environments can be found in the Integration sample folder. By default the location of the Integration folder is the following:

..\Program Files\EuroPlus\NiceLabel v5 Integration Documents and Samples\

The ActiveX integration samples can also be downloaded from http://www.nicelabel.com/downloads/dwld_samples.php.

6.8 Can I Control NiceForm through the ActiveX Interface?

NiceForm has two ActiveX interfaces. Interfaces are divided in the design and run mode. Design mode is useful for designing the form on several ways and run mode is useful only for form execution directly from your software.

For more information refer to the NiceForm ActiveX interfaces chapter.

7. Glossary

ActiveX Controls	An ActiveX Control is a stand-alone software component who's function is pre-defined and performs in a standard way. It exemplifies component software. The ActiveX Control specification defines this standard and sets the stage for plug-in component architecture. An ActiveX Control is a component of functionality that can be purchased and added to a Windows application. In essence, an entire application can now be built from pre-existing parts (reusable software components). Bear in mind that ActiveX Controls are not, by themselves, applications. They are service providers (servers) that plug into a control container. As with other ActiveX technologies, interactions between participating software units are specified by various interfaces supported by COM. In the case of ActiveX controls, the involved software units are the control and its container. An ActiveX Control may incorporate many other ActiveX technologies, all of which are COM-based. As an example, controls often support embedding interfaces as well as automation, allowing access to their methods.
Automation	Automation is a means in which a developer can access an application's functionality through COM-object support. This is not source code exposure, but component exposure. Automation provides programmers with the means to pull a feature from an existing application and add the same feature to their own project. For instance, since Microsoft Excel supports automation, programmers can incorporate existing Excel features into their work. With automation, Excel is no longer just an end-user application, it has become a valuable, programming toolkit.
Client/Server architecture	A network architecture in which each computer or process on the network is either a client or a server. Servers are powerful computers or processes dedicated to managing disk drives (file servers), printers (print servers), or network traffic (network servers). Clients are PCs or workstations on which users run applications. Clients rely on servers for resources such as files, devices, and even processing power. Another type of network architecture is known as a *peer-to-peer* architecture because each node has equivalent responsibilities. Both client/server and peer-to peer architectures are widely used and each has unique advantages and disadvantages. Client/server architectures are sometimes called *two-tier architectures.*
Compound Documents	Compound document technology addresses the need for integration between applications, effectively allowing different applications to work together smoothly, producing what appears to be a single document. It

	allows using existing applications to work in various portions of a document, totally independent of each other. As previously mentioned, OLE defines the interface standards and allows this kind of interaction between a wide variety of applications and vendors.
Distributed COM	Distributed COM (DCOM) is mechanism for COM objects to provide their services across machine boundaries. Originally, COM implementation was restricted to a single machine. COM objects could be implemented on the same machine as their client but they couldn’t reside on other machines in the network. DCOM removes this restriction with the remote procedure call (RPC), allowing a client to execute an object across a network. DCOM also provides support for security services by controlling which clients can use a COM object
DLL	Short for *Dynamic Link Library,* a library of executable functions or data that can be used by a Windows application. Typically, a DLL provides one or more particular functions and a program accesses the functions by creating either a static or dynamic link to the DLL. A static link remains constant during program execution while a dynamic link is created by the program as needed. DLLs can also just contain data. DLL files usually end with the extension *.dll, .exe, .drv,* or *.fon.* A DLL can be used by several applications at the same time. Some DLLs are provided with the Windows operating system and are available for any Windows application. Other DLLs are written for a particular application and are loaded with the application.
Property	A property is an attribute of an object that defines one of the object's characteristics, such as size, color, or screen location, or an aspect of its behavior, such as whether it is enabled or visible. To change the characteristics of an object, you change the values of its properties. To set the value of a property, follow the reference to an object with a period, the property name, an equal sign (=), and the new property value.
Object	An object represents an element of an application, such as a worksheet, a cell, a chart, a form, or a report. In Visual Basic code, you must identify an object before you can apply one of the object’s methods or change the value of one of its properties.
Component Model	A ***component model*** defines one or more required component interfaces, allowable patterns of interactions among components, communication behaviors among components and between components and the component runtime system, and, possibly, a programming model for component developers.

8. Contacts

http://www.nicelabel.com/Contact