ActiveX and Dynamic Data Exchange Programmers Interface

White Paper

Version 021227-01
# Table of Contents

1. **Introduction** ........................................................................................................................................................................ 4
   1.1 **Overview** ...................................................................................................................................................................... 4
   1.2 **Introduction to DDE and ActiveX** ........................................................................................................................... 4
      1.2.1 What is Dynamic Data Exchange (DDE)? .............................................................................................................. 4
      1.2.2 What is COM / ActiveX? ........................................................................................................................................ 5
      1.2.3 What is a Type Library? ...................................................................................................................................... 5
      1.2.4 The OLE/COM Object Viewer ............................................................................................................................. 6
   1.3 **What is Visual Basic** ..................................................................................................................................................... 6

2. **DDE Communication** ........................................................................................................................................................... 7
   2.1 **Introduction** .................................................................................................................................................................... 7
   2.2 **DDE Commands** ............................................................................................................................................................ 7
      2.2.1 DDEInitiate .............................................................................................................................................................. 7
      2.2.2 DDEExecute .......................................................................................................................................................... 7
      2.2.3 DDETerminate ...................................................................................................................................................... 8
   2.3 **GalleryCommands** ......................................................................................................................................................... 8
      2.3.1 LABEL .................................................................................................................................................................... 8
      2.3.2 FILECLOSE ......................................................................................................................................................... 8
      2.3.3 LABELCLOSE .................................................................................................................................................... 9
      2.3.4 SET ........................................................................................................................................................................ 9
      2.3.5 COMMENT .......................................................................................................................................................... 9
      2.3.6 PRINT .................................................................................................................................................................. 9
      2.3.7 PORT ..................................................................................................................................................................10
      2.3.8 PRINTER ............................................................................................................................................................10
      2.3.9 PRINTJOBNAME ............................................................................................................................................10
      2.3.10 LOGIN ..............................................................................................................................................................11
      2.3.11 RETURN ..........................................................................................................................................................11
      2.3.12 QUIT ................................................................................................................................................................ 11
      2.3.13 MESSAGEBOX ................................................................................................................................................11
      2.3.14 CREATEFILE ..................................................................................................................................................12
      2.3.15 DELETEFILE ..................................................................................................................................................12
      2.3.16 SESSIONSTART ...........................................................................................................................................12
      2.3.17 SESSIONPRINT ........................................................................................................................................12
      2.3.18 SESSIONEND ................................................................................................................................................13
      2.3.19 OEMTOANSI ................................................................................................................................................13
      2.3.20 SETDATABASE ...............................................................................................................................................13
      2.3.21 SETTABLE ....................................................................................................................................................14

3. **ActiveX interface** ................................................................................................................................................................. 15
   3.1 **Introduction** ................................................................................................................................................................. 15
   3.2 **LabelGallery ActiveX interface version 1** .................................................................................................................. 15
      3.2.1 Class LabelGallery .............................................................................................................................................. 15
   3.3 **LabelGallery ActiveX interface version 2** .................................................................................................................. 23
      3.3.1 Hierarchy diagram ................................................................................................................................................ 23
      3.3.2 Class IBarcode ................................................................................................................................................... 25
      3.3.3 Class IDatabase ............................................................................................................................................... 36
      3.3.4 Class IDatabaseList ...................................................................................................................................... 41
      3.3.5 Class IDBDef .................................................................................................................................................... 43
      3.3.6 Class IDBField ............................................................................................................................................... 44
      3.3.7 Class IDBParameter ........................................................................................................................................ 46
3.3.8  Class IEXTFunction .............................................................................................................................................. 47
3.3.9  Class IFieldList .................................................................................................................................................. 50
3.3.10 Class IFunction .................................................................................................................................................... 51
3.3.11 Class IFunctionList ........................................................................................................................................... 54
3.3.12 Class IGraphics ................................................................................................................................................... 55
3.3.13 Class ILabel ......................................................................................................................................................... 64
3.3.14 Class ILabelGallery .......................................................................................................................................... 70
3.3.15 Class IObject ....................................................................................................................................................... 79
3.3.16 Class IObjectList ............................................................................................................................................... 87
3.3.17 Class IOleObject ................................................................................................................................................. 87
3.3.18 Class IParagraph ............................................................................................................................................... 95
3.3.19 Class IParameterList ....................................................................................................................................... 107
3.3.20 Class IRectangle .............................................................................................................................................. 108
3.3.21 Class IRTFText ................................................................................................................................................... 117
3.3.22 Class IText ......................................................................................................................................................... 125
3.3.23 Class IVar ......................................................................................................................................................... 136
3.3.24 Class IVariableList ......................................................................................................................................... 149

4.  Programming Samples .................................................................................................................................................. 153

4.1  DDE ............................................................................................................................................................................. 153
4.2  ActiveX interface ......................................................................................................................................................... 153
4.2.1 ActiveX Hints ........................................................................................................................................................... 153

5.  Glossary ......................................................................................................................................................................... 157

5.1  ActiveX Controls ........................................................................................................................................................ 157
5.2  Automation .................................................................................................................................................................. 157
5.3  Client/Server .............................................................................................................................................................. 157
5.4  Compound Documents ............................................................................................................................................. 158
5.5  Distributed COM ........................................................................................................................................................ 158
5.6  DLL ............................................................................................................................................................................... 158
5.7  Property ...................................................................................................................................................................... 158
5.8  Object .......................................................................................................................................................................... 158
5.9  Component model ...................................................................................................................................................... 159
1. Introduction

1.1 Overview

The most common method of printing labels is directly from LabelGallery. But sometimes there might be other requirements for label production. LabelGallery has a wide variety of connectivity and integration options so you do not need to use LabelGallery interactively, but thru an ActiveX interface or Dynamic Data Exchange connectivity. LabelGallery can be used as a "print-engine" totally integrated to your custom application and invisible to the end-user's eyes. The purpose of this manual is to help you program your own application to control the LabelGallery labeling software. All you should know about integrating LabelGallery into your application is described in this manual. The information in this chapter is for advanced users and application developers only. If you do not plan to write applications that use LabelGallery to print labels, you can skip reading this manual entirely. For more information about ActiveX and Dynamic Data Exchange, please read following chapter.

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 the 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 nondynamic 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 multiplatform 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. Often in documentation, COM and ActiveX are used interchangeably. 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 a .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, which may be downloaded from the Microsoft web site.

The type library supplied by the LabelGallery labeling software is a file called LGPLUS.olb that describes the type of all of the ActiveX objects.

The type library does not store objects, 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 the names and parameters. This library helps you to write your program because it contains all of the definitions of object methods and properties that you can access. Using this library you optimize your job.

The name of this type library is LabelGalleryPlus.

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

To install the type library:

- Choose Project References.
• Activate LabelGallery in the list of available references then validate the dialog box.

**To display the methods and properties:**

• Use the Object Explorer
• In the library list select LabelGallery

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.

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 LabelGallery type library contents. Use this to figure out what methods, properties, and events LabelGallery supports!

To create the LabelGallery type library go to the command prompt and type the following command (you have to be in the folder “C:\Program Files\Sato\Gallery\Bin” where LabelGallery’s executable LGPLUS.EXE is stored)

```bash
LGPLUS /typelib
```

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

OLE/COM Object Viewer is available to download from:


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.
2. DDE Communication

2.1 Introduction

To create the DDE communication, the client application must use the following DDE parameters:

Service = LabelGallery
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 LabelGallery commands.

When you want to use DDE communication to control LabelGallery, 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:

LGPLUS.EXE <label_name> /s

The parameter s (silent) prevents LabelGallery from being visible on the screen. It will run in a minimized form.

2.2 DDE Commands

2.2.1 DDEInitiate

*Description:*
DDEInitiate command initiates a link between LabelGallery and your application.

*Syntax:*
DDEInitiate(Service, Topic)

2.2.2 DDEExecute

*Description:*
DDEExecute command executes a command on your label. For list of available commands please see GalleryCommands chapter of this manual.
2.2.3 DDETerminate

**Description:**
DDETerminate command terminates a link between LabelGallery and your application.

**Syntax:**
DDETerminate Channel

2.3 Gallery Commands

Printing with LabelGallery can be accomplished automatically. There are two 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. The second 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.

2.3.1 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.

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\SATO\Labels\sample3.lbl").

If you use the LABEL command with GalleryWatch 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>

2.3.2 FILECLOSE

**Description:**
The FILECLOSE command closes the currently active label. LabelGallery will stay opened.

**Syntax:**
FILECLOSE
2.3.3 LABELCLOSE

**Description:**
The LABELCLOSE command closes the currently active label. LabelGallery will stay opened.
This command is introduced as a synonym for FILECLOSE command.

**Syntax:**
LABELCLOSE

2.3.4 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]]

2.3.5 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 LabelGallery.

**Syntax:**
;

2.3.6 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]

### 2.3.7 PORT

**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>

### 2.3.8 PRINTER

**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 LabelGallery 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>

### 2.3.9 PRINTJOBNAME

**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>

### 2.3.10 LOGIN

**Description:**
Performs login procedure into LabelGallery program. This is necessary when login into LabelGallery is required.

**NOTE.** This is DDE command and cannot be used in .JOB files.

**Syntax:**
LOGIN <username>

### 2.3.11 RETURN

**Description:**
This command returns to the LabelGallery program after printing.

**Syntax:**
Return

### 2.3.12 QUIT

**Description:**
This command stops the LabelGallery program after printing. The application is closed.

**Syntax:**
Quit

### 2.3.13 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]
2.3.14 CREATEFILE

**Description:**
This command lets you create the file with the contents LabelGallery. 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 LabelGallery 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>

2.3.15 DELETEFILE

**Description:**
This command deletes the file.

See also:
CREATEFILE

**Syntax:**
DELETEFILE <name_of_the_file>

2.3.16 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 GalleryCommands and it is not necessary to generate a complete data stream for each print session.

See also:
SESSIONPRINT
SESSIONEND
2.3.17 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 Gallery Command PRINT.

See also:
PRINT
SESSIONSTART
SESSIONEND

2.3.18 SESSIONEND

**Description:**

The function closes data stream.

See also:
SESSIONSTART
SESSIONPRINT

2.3.19 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 LabelGallery at print time.

**Syntax:**

OEMTOANSI ON|OFF

2.3.20 SETDATABASE

**Description:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>database_name</td>
<td>the name of the currently used database as defined in the program</td>
</tr>
<tr>
<td>value</td>
<td>name of the new table that should be used as data source</td>
</tr>
</tbody>
</table>
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>

### 2.3.21 SETTABLE

**Description:**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td>the name of the currently used table as defined in the program</td>
</tr>
<tr>
<td>value</td>
<td>name of the new table that should be used as data source</td>
</tr>
</tbody>
</table>

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>
3. ActiveX interface

3.1 Introduction

LabelGallery Plus version 1.2 introduced a new programming interface and at the same time all previous programming methods are still available.

What has been added is entirely new interface with new commands and new automation possibilities. Updated commands actually allow you to have more control over LabelGallery from your own application. And it does not stand just for printing process, but also for design part that can now be part of your application.

Previous version of Automation allowed a limited insight to label's structure. You could only set the variable values, then initiate printing and that was it. With DDE communication no information is returned to the application if the variable assignment was successful or not. Automation (Active X) allows also the status of variable managing procedure to be returned to your application so you can control printing process more accurately.

Enhanced ActiveX programming interface brings improvement over the previous implementation. Besides variable setting a lot more programming functionality is allowed. New ActiveX interface makes it possible to query every label element for its properties. The same goes for variables and functions defined on the label. The properties of all label elements can be modified prior printing if for some reason you do not want to use label-defined settings. You can even create a label preview in your own application.

The best example of new interface use is LabelGallery Plus. From version 1.2 Label Inspector component is part of LabelGallery software. It is not an integral part of the LabelGallery application, but is additional component that connects to LabelGallery entirely thru ActiveX programming interface version 2. Label Inspector is a toolbar in LabelGallery, you can enable it in View menu.

3.2 LabelGallery ActiveX interface version 1

3.2.1 Class LabelGallery

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>CloseForm</td>
<td></td>
</tr>
<tr>
<td>DetailedMessage</td>
<td></td>
</tr>
<tr>
<td>ErrorID</td>
<td></td>
</tr>
</tbody>
</table>
Quit

**Description:**
This procedure closes the LabelGallery if it was opened with the client application.

**Syntax:**
Quit

Login

**Description:**
Performs login into LabelGallery if it has not been executed already.

**Syntax:**
Function Login(UserName As String, Level As Long) As Boolean

**JobRun**

**Description:**
Function executes specified JOB file.

**Syntax:**
Function JobRun(FileName As String) As Boolean

**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.

**Syntax:**
Function LabelOpen(FileName As String) As Long

**LabelClose**

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

**Syntax:**
Function LabelClose(LabelID As Long) As Boolean

**LabelPrint**

**Description:**
This function prints the label with the specified label ID. Quantity is a string, where the reserved words VARIABLE or UNLIMITED can also be specified. Otherwise the quantity of the labels is specified.

After quantity, you can also specify number of labels to skip on page. Syntax is "Quantity, Skip".

**Syntax:**
Function LabelPrint(LabelID As Long, Quantity As String) As Boolean
**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`

**LabelGetVarCount**

*Description:*
Function returns number of variables on the label with specified label ID.

*Syntax:*
Function `LabelGetVarCount(LabelID As Long) As Long`

**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, Var As Long) As String`

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>ID</th>
<th>Used</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VarQuantity</td>
<td>inputType</td>
<td>Description</td>
<td>prefix</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
<td>--------</td>
</tr>
</tbody>
</table>

InputType has the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prompt</td>
</tr>
<tr>
<td>2</td>
<td>System clock</td>
</tr>
<tr>
<td>3</td>
<td>PrinterClock</td>
</tr>
<tr>
<td>4</td>
<td>Global</td>
</tr>
<tr>
<td>5</td>
<td>Generated</td>
</tr>
<tr>
<td>6</td>
<td>Database</td>
</tr>
</tbody>
</table>

For prompted variables:

"Prompt"
"PromptType"
"DefValue"
"DefType"
"IsDynamic"
"IsRequired"

Variable types are self-explanatory. If the type is bool, 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

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

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

ExecuteMacro

Description:
This function executes a Macro (DDE) command.
**Syntax:**
Function ExecuteMacro(LabelID As Long, Macro As String) As Boolean

**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

**LabelSetPrintJobName**

*Description:*
Using this function you can set the Print job name that LabelGallery 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

**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 LabelGallery 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

**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

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.

Syntax:
Function LabelSessionStart(LabelID As Long) As Boolean

See also:
LabelSessionPrint
LabelSessionEnd

LabelSessionPrint

Description:
You send the data stream to printer using this function. 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

LabelSessionEnd

Description:
The function ends print session.

Syntax:
Function LabelSessionEnd(LabelID As Long) As Boolean

See also:
LabelSessionStart
LabelSessionPrint

3.3 LabelGallery ActiveX interface version 2

3.3.1 Hierarchy diagram
The diagram below shows the connection between all interfaces included in LGPLUS.olb:
3.3.2 Class IBarcode

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Left</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>Top</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Width</td>
<td>AutoCDCalculation</td>
</tr>
<tr>
<td>Height</td>
<td>IncludeCD</td>
</tr>
<tr>
<td>Kind</td>
<td>HasQuietZone</td>
</tr>
<tr>
<td>Status</td>
<td>PrintAsGraphics</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>PageNumber</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>Variable</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Move</td>
<td>Rotation</td>
</tr>
<tr>
<td>Resize</td>
<td>Selected</td>
</tr>
<tr>
<td>SetVariable</td>
<td>Name</td>
</tr>
<tr>
<td>Contents</td>
<td>ZOrder</td>
</tr>
<tr>
<td>BarcodeType</td>
<td></td>
</tr>
<tr>
<td>SetContents</td>
<td></td>
</tr>
<tr>
<td>GetProperty</td>
<td></td>
</tr>
</tbody>
</table>

**AnchorElementID**

*Description:*
Currently not used. Intended to be used, when position of one element can be dependent 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
<tr>
<td>8</td>
<td>bottom right</td>
</tr>
</tbody>
</table>

*Syntax:*
AnchorPoint As Long

**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, LabelGallery will verify the contents of the barcode, as the checkdigit is provided together with the data.

*Syntax:*
AutoCDCalculation As Boolean

**IncludeCD**

*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
**HasQuietZone**

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

*Syntax:*
HasQuietZone As Boolean

**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

**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

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
</tbody>
</table>
Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

**Syntax:**

FormatID As Long

---

### IsLocked

**Description:**

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

**Syntax:**

IsLocked As Boolean

---

### 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

**Syntax:**

Rotation 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

**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

**ID**

*Description:*
ID of the element

*Syntax:*
Property ID 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

**Top**

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

**Syntax:**
Property Top As Long

**Access Rights:**
read-only

**Width**

**Description:**
Width of the element (in 0.01 mm units).

**Syntax:**
Property Width As Long

**Access Rights:**
read-only

**Height**

**Description:**
Height of the element (in 0.01 mm units).

**Syntax:**
Property Height As Long

**Access Rights:**
read-only
**Kind**

*Description:*
Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
<tr>
<td>307</td>
<td>OleObject</td>
</tr>
<tr>
<td>308</td>
<td>Downloaded Graphic Object</td>
</tr>
<tr>
<td>309</td>
<td>ParagraphObject</td>
</tr>
<tr>
<td>312</td>
<td>RTFTextObject</td>
</tr>
<tr>
<td>313</td>
<td>EllipseObject</td>
</tr>
</tbody>
</table>

*Syntax:*
Property Kind As Long

*Access Rights:*
read-only

**Status**

*Description:*
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>
**Syntax:**
Property Status As Long

**Access Rights:**
read-only

**RotateFlag**

**Description:**
Flag, which defines, how the object can be rotated.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**
Property RotateFlag As Long

**Access Rights:**
read-only

**ResizeFlag**

**Description:**
Flag, which defines, how the object can be resized.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

**Syntax:**
Property ResizeFlag As Long
Access Rights:
read-only

**Variable**

*Description:*
Returns the interface to the variable, which is attached to the element.

*Syntax:*
Property Variable As IVar

*Access Rights:*
read-only

**Move**

*Description:*
Move the element to the location X, Y

*Syntax:*
Move(X As Long, Y 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)

**SetVariable**

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

*Syntax:*
**Function** SetVariable(ID As Long) As Long

**Contents**

**Description:**
Returns the current contents of the element.

**Syntax:**
Property Contents As String

**Access Rights:**
read-only

**BarcodeType**

**Description:**
Returns the Barcode Type.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EAN13</td>
</tr>
<tr>
<td>2</td>
<td>EAN8</td>
</tr>
<tr>
<td>3</td>
<td>UPC_A</td>
</tr>
<tr>
<td>4</td>
<td>UPC_E</td>
</tr>
<tr>
<td>5</td>
<td>I2OF5</td>
</tr>
<tr>
<td>6</td>
<td>CODE39</td>
</tr>
<tr>
<td>7</td>
<td>CODE128A</td>
</tr>
<tr>
<td>8</td>
<td>CODE128B</td>
</tr>
<tr>
<td>9</td>
<td>CODE128C</td>
</tr>
<tr>
<td>10</td>
<td>CODABAR</td>
</tr>
<tr>
<td>11</td>
<td>CODE128</td>
</tr>
<tr>
<td>12</td>
<td>UCC128</td>
</tr>
<tr>
<td>13</td>
<td>POSTNET32</td>
</tr>
<tr>
<td>14</td>
<td>POSTNET37</td>
</tr>
<tr>
<td>15</td>
<td>POSTNET52</td>
</tr>
<tr>
<td>16</td>
<td>POSTNET62</td>
</tr>
<tr>
<td>17</td>
<td>BOOKLAND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>18</td>
<td>DISTRIBUTION</td>
</tr>
<tr>
<td>19</td>
<td>CODE93</td>
</tr>
<tr>
<td>20</td>
<td>NG_EAN13</td>
</tr>
<tr>
<td>21</td>
<td>NG_EAN8</td>
</tr>
<tr>
<td>22</td>
<td>PDF417</td>
</tr>
<tr>
<td>23</td>
<td>BC412</td>
</tr>
<tr>
<td>24</td>
<td>DATAMATRIX</td>
</tr>
<tr>
<td>25</td>
<td>MAXICODE</td>
</tr>
<tr>
<td>26</td>
<td>EAN128</td>
</tr>
<tr>
<td>27</td>
<td>AZTEC</td>
</tr>
<tr>
<td>28</td>
<td>QR</td>
</tr>
<tr>
<td>29</td>
<td>SSCC</td>
</tr>
<tr>
<td>30</td>
<td>MSI</td>
</tr>
<tr>
<td>31</td>
<td>CBLOCK_F</td>
</tr>
<tr>
<td>32</td>
<td>PHARMACODE</td>
</tr>
<tr>
<td>33</td>
<td>EXTCODE39</td>
</tr>
<tr>
<td>34</td>
<td>EAN14</td>
</tr>
<tr>
<td>35</td>
<td>KIX</td>
</tr>
<tr>
<td>36</td>
<td>ITF14</td>
</tr>
<tr>
<td>37</td>
<td>DPHARMACODE</td>
</tr>
<tr>
<td>38</td>
<td>CODE32</td>
</tr>
<tr>
<td>39</td>
<td>EAN13+2</td>
</tr>
<tr>
<td>40</td>
<td>EAN13+5</td>
</tr>
<tr>
<td>41</td>
<td>EAN8+2</td>
</tr>
<tr>
<td>42</td>
<td>EAN8+5</td>
</tr>
<tr>
<td>43</td>
<td>UPC_A+2</td>
</tr>
<tr>
<td>44</td>
<td>UPC_A+5</td>
</tr>
<tr>
<td>45</td>
<td>UPC_E+2</td>
</tr>
<tr>
<td>46</td>
<td>UPC_E+5</td>
</tr>
<tr>
<td>47</td>
<td>ITF16</td>
</tr>
<tr>
<td>48</td>
<td>MICROPDF</td>
</tr>
</tbody>
</table>

**Syntax:**

Property BarcodeType As String

**Access Rights:**

read-only
GetProperty

Description:
Returns the property of a barcode.
Possible names are:

IsStructured – used for MaxiCode barcode

Syntax:
Function GetProperty(Name As String) As String

3.3.3 Class IDatabase

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Kind</td>
<td>Description</td>
</tr>
<tr>
<td>InputVars</td>
<td>ReviewBeforePrint</td>
</tr>
<tr>
<td>OutputVars</td>
<td>DriverType</td>
</tr>
<tr>
<td>Fields</td>
<td>Alias</td>
</tr>
<tr>
<td>Parameters</td>
<td>IsDelimited</td>
</tr>
<tr>
<td></td>
<td>Delimiter</td>
</tr>
<tr>
<td></td>
<td>Separator</td>
</tr>
<tr>
<td></td>
<td>DBPassword</td>
</tr>
<tr>
<td></td>
<td>Table</td>
</tr>
<tr>
<td></td>
<td>Order</td>
</tr>
<tr>
<td></td>
<td>Navigation</td>
</tr>
<tr>
<td></td>
<td>SQL</td>
</tr>
</tbody>
</table>

Name

Description:
In the Name property name of the database is stored. This property can be set in LabelGallery Plus on General tab of Database dialog box.

Syntax:
Name As String
Description

Description:
In the Description property description of the database is stored. This property can be set in LabelGallery Plus on General tab of Database dialog box.

Syntax:
Description As String

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

DriverType

Description:
DriverType is in main driver name. You can find suitable names in BDE Administrator and also in LabelGallery Plus in Database dialog box under the driver selection.

Syntax:
DriverType As String

Alias

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

Syntax:
Alias 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

---

**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

---

**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

---

**DBPassword**

**Description:**

DBPassword property represents the password for the database.

**Syntax:**

DBPassword As String

---

**Table**

**Description:**

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

Order

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

Syntax:
Order As String

Navigation

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

Possible values for the Navigation property are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>First</td>
</tr>
<tr>
<td>1</td>
<td>Select</td>
</tr>
<tr>
<td>2</td>
<td>All</td>
</tr>
<tr>
<td>3</td>
<td>Last</td>
</tr>
</tbody>
</table>

Syntax:
Navigation As Long

SQL

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
**ID**

*Description:*  
ID of the database.

*Syntax:*  
Property ID As Long

*Access Rights:*  
read-only

**Kind**

*Description:*  
Kind property for the database is always set to 4.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Concatenate</td>
</tr>
<tr>
<td>2</td>
<td>Subset</td>
</tr>
<tr>
<td>3</td>
<td>Linear Function</td>
</tr>
<tr>
<td>4</td>
<td>Database</td>
</tr>
<tr>
<td>5</td>
<td>EAN 128</td>
</tr>
<tr>
<td>6</td>
<td>CD Algo</td>
</tr>
<tr>
<td>7</td>
<td>Date Addition</td>
</tr>
<tr>
<td>8</td>
<td>Euro</td>
</tr>
<tr>
<td>9</td>
<td>External</td>
</tr>
<tr>
<td>10</td>
<td>SSCC</td>
</tr>
</tbody>
</table>

*Syntax:*  
Property Kind 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

**OutputVars**

**Description:**
Returns the interface to the variable list for output variables.

**Syntax:**
Property OutputVars As IVariableList

**Access Rights:**
read-only

**Fields**

**Description:**
Returns the interface to the fields which are present in the database.

**Syntax:**
Property Fields As IFieldList

**Access Rights:**
read-only

**Parameters**

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

**Syntax:**
Property Parameters As IParameterList

**Access Rights:**
read-only

**3.3.4 Class IDatabaseList**
<table>
<thead>
<tr>
<th>Count</th>
<th>&lt;none&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>FindByName</td>
<td></td>
</tr>
<tr>
<td>FindByID</td>
<td></td>
</tr>
</tbody>
</table>

**Count**

*Description:*
Count property returns 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

**FindByID**

*Description:*
Returns the interface to the database. Database is selected with database ID.

*Syntax:*
Function FindByID(ID As Long) As IDatabase
3.3.5 Class IDBDef

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;none&gt;</td>
<td>DriverType</td>
</tr>
<tr>
<td></td>
<td>Alias</td>
</tr>
<tr>
<td></td>
<td>IsDelimited</td>
</tr>
<tr>
<td></td>
<td>Separator</td>
</tr>
<tr>
<td></td>
<td>DBPassword</td>
</tr>
</tbody>
</table>

**DriverType**

*Description:*
DriverType is in main driver name. You can find suitable names in BDE Administrator and also in LabelGallery Plus in Database dialog box under the driver selection.

*Syntax:*
DriverType As String

**Alias**

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

*Syntax:*
Alias 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

**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

**DBPassword**

**Description:**
DBPassword property represents the password for the database.

**Syntax:**
DBPassword As String

### 3.3.6 Class IDBField

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>IsSelected</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td></td>
</tr>
</tbody>
</table>

**Number**

**Description:**
Number property represents the number of the field.

**Syntax:**
Property Number 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

**Length**

**Description:**
Length property represents the length of the field.

**Syntax:**
Property Length As Long

**Access Rights:**
read-only

**Offset**

**Description:**
Offset property represents the offset of the beginning of the line. This property is available only for text files.

**Syntax:**
Property Offset As Long

**Access Rights:**
read-only

**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
Variable

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

Syntax:
Property Variable As IVar

Access Rights:
read-only

3.3.7 Class IDBParameter

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>SetVariable</td>
<td>Field</td>
</tr>
<tr>
<td></td>
<td>Relation</td>
</tr>
<tr>
<td></td>
<td>Variable</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>==</td>
</tr>
<tr>
<td>1</td>
<td>&gt;</td>
</tr>
<tr>
<td>2</td>
<td>&gt;=</td>
</tr>
<tr>
<td>3</td>
<td>&lt;</td>
</tr>
<tr>
<td>4</td>
<td>&lt;=</td>
</tr>
<tr>
<td>5</td>
<td>&lt;&gt;</td>
</tr>
</tbody>
</table>
**Syntax:**
Relation As Long

**Variable**

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

**Syntax:**
Property Variable As IVar

**Access Rights:**
read-only

**SetVariable**

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

**Syntax:**
Function SetVariable(ID As Long) As Long

### 3.3.8 Class IEXTFunction

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Kind</td>
<td>Description</td>
</tr>
<tr>
<td>InputVars</td>
<td>ReviewBeforePrint</td>
</tr>
<tr>
<td>OutputVars</td>
<td>Definition</td>
</tr>
<tr>
<td>ProviderName</td>
<td></td>
</tr>
<tr>
<td>IsContentsProvider</td>
<td></td>
</tr>
</tbody>
</table>
**Name**

*Description:*
Name property represents the name of the function.

*Syntax:*
Name As String

**Description**

*Description:*
In the Description property description of the function is stored.

*Syntax:*
Description As String

**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

**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

**ID**

*Description:*
ID of the element

*Syntax:*

Property ID As Long

**Access Rights:**
read-only

**Kind**

**Description:**
This property represents the value for the type of the function.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Concatenate</td>
</tr>
<tr>
<td>2</td>
<td>Subset</td>
</tr>
<tr>
<td>3</td>
<td>Linear Function</td>
</tr>
<tr>
<td>4</td>
<td>Database</td>
</tr>
<tr>
<td>5</td>
<td>EAN 128</td>
</tr>
<tr>
<td>6</td>
<td>CD Algo</td>
</tr>
<tr>
<td>7</td>
<td>Date Addition</td>
</tr>
<tr>
<td>8</td>
<td>Euro</td>
</tr>
<tr>
<td>9</td>
<td>External</td>
</tr>
<tr>
<td>10</td>
<td>SSCC</td>
</tr>
</tbody>
</table>

**Syntax:**
Property Kind 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:**
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

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

3.3.9 Class IFieldList

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>&lt;none&gt;</td>
</tr>
</tbody>
</table>
### Count

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

**Syntax:**
Property Count As Long

**Access Rights:**
read-only

### Item

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

**Syntax:**
Function Item(Index As Long) As IDBField

### 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

### 3.3.10 Class IFunction

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Kind</td>
<td>Description</td>
</tr>
<tr>
<td>InputVars</td>
<td>ReviewBeforePrint</td>
</tr>
<tr>
<td>OutPutVars</td>
<td></td>
</tr>
</tbody>
</table>
**Name**

*Description:*
Name property represents the name of the function.

*Syntax:*
Name As String

**Description**

*Description:*
In the Description property description of the function is stored.

*Syntax:*
Description As String

**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

**ID**

*Description:*
ID of the element

*Syntax:*
Property ID As Long

*Access Rights:*
read-only

**Kind**

*Description:*

This property represents the value for the type of the function. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Concatenate</td>
</tr>
<tr>
<td>2</td>
<td>Subset</td>
</tr>
<tr>
<td>3</td>
<td>Linear Function</td>
</tr>
<tr>
<td>4</td>
<td>Database</td>
</tr>
<tr>
<td>5</td>
<td>EAN 128</td>
</tr>
<tr>
<td>6</td>
<td>CD Algo</td>
</tr>
<tr>
<td>7</td>
<td>Date Addition</td>
</tr>
<tr>
<td>8</td>
<td>Euro</td>
</tr>
<tr>
<td>9</td>
<td>External</td>
</tr>
<tr>
<td>10</td>
<td>SSCC</td>
</tr>
</tbody>
</table>

**Syntax:**

Property Kind 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

**OutputVars**

**Description:**
Returns the interface to the variable list for output variables.

**Syntax:**
Property OutputVars As IVariableList

**Access Rights:**
read-only

### 3.3.11 Class IFunctionList

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>FindByName</td>
<td></td>
</tr>
<tr>
<td>FindByID</td>
<td></td>
</tr>
<tr>
<td>CreateExternalFunction</td>
<td></td>
</tr>
</tbody>
</table>

**Count**

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

**Syntax:**
Property Count As Long

**Access Rights:**
read-only

**Item**

**Description:**
Returns the interface to the function. Function is selected with Index.

**Syntax:**
Function Item(Index As Long) As IFunction

**FindByName**

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

**Syntax:**
Function FindByName(Name As String) As IFunction

**FindByID**

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

**Syntax:**
Function FindByID(ID As Long) As IFunction

**CreateExternalFunction**

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

**Syntax:**
Function CreateExternalFunction() As IExtFunction

### 3.3.12 Class IGraphics

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Heigth</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>Zorder</td>
</tr>
<tr>
<td>Variable</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Move</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Resize</td>
<td>HorzMirror</td>
</tr>
<tr>
<td>SetVariable</td>
<td>VertMirror</td>
</tr>
<tr>
<td>Load</td>
<td>OnMemoryCard</td>
</tr>
<tr>
<td></td>
<td>ResizeMode</td>
</tr>
<tr>
<td></td>
<td>FileName</td>
</tr>
</tbody>
</table>
Name

*Description:*
Name property represents the name of the graphics.

*Syntax:*
Name As String

Selected

*Description:*
When the element is selected, this property is TRUE.

*Syntax:*
Selected As Boolean

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

*Syntax:*
Rotation As Long

IsLocked

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

**Syntax:**
IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
</tbody>
</table>
Syntax:
AnchorPoint As Long

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

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

Syntax:
FormatID As Long

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**

*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:*
```plaintext```
PageNumber As Long
```

**HorzMirror**

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

*Syntax:*
```plaintext```
HorzMirror As Boolean
```

**VertMirror**

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

*Syntax:*
```plaintext```
VertMirror As Boolean
```

**OnMemoryCard**

*Description:*
In case that graphics is stored on a memory card then OnMemoryCard property has value TRUE.

*Syntax:*
```plaintext```
OnMemoryCard As Boolean
```

**ResizeMode**

*Description:*
With ResizeMode it is possible to set the way of resizing the graphics.
Possible values for ResizeMode are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Original size</td>
</tr>
<tr>
<td>1</td>
<td>Proportional resize</td>
</tr>
<tr>
<td>2</td>
<td>Fit to size.</td>
</tr>
</tbody>
</table>

**Syntax:**

ResizeMode As Long

**ID**

**Description:**
ID of the element

**Syntax:**

Property ID 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

**Top**

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

**Syntax:**

Property Top As Long
**Access Rights:**
read-only

**Width**

**Description:**
Width of the element (in 0.01 mm units).

**Syntax:**
Property Width As Long

**Access Rights:**
read-only

**Height**

**Description:**
Height of the element (in 0.01 mm units).

**Syntax:**
Property Height As Long

**Access Rights:**
read-only

**Status**

**Description:**
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

**Syntax:**
Property Status As Long
**Access Rights:**
read-only

**RotateFlag**

**Description:**
Flag, which defines, how the object can be rotated.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**
Property RotateFlag As Long

**Access Rights:**
read-only

**ResizeFlag**

**Description:**
Flag, which defines, how the object can be resized.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

**Syntax:**
Property ResizeFlag 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

Move

*Description*:  
Move the element to the location X, Y

*Syntax*:  
Move(X As Long, Y As Long)

SetVariable

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

*Syntax*:  
Function SetVariable(ID As Long) As Long

FileName

*Description*:  
FileName property represents the file name of the graphics file.

*Syntax*:  
Property FileName As String

*Access Rights*:  
read-only
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

3.3.13 Class ILabel

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objects</td>
<td>PrinterName</td>
</tr>
<tr>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td>Functions</td>
<td></td>
</tr>
<tr>
<td>Databases</td>
<td></td>
</tr>
<tr>
<td>CurrUnit</td>
<td></td>
</tr>
<tr>
<td>GlobalVariables</td>
<td></td>
</tr>
<tr>
<td>NewVariable</td>
<td></td>
</tr>
<tr>
<td>NewFunction</td>
<td></td>
</tr>
<tr>
<td>NewDatabase</td>
<td></td>
</tr>
<tr>
<td>ObjectProperties</td>
<td></td>
</tr>
<tr>
<td>VariableProperties</td>
<td></td>
</tr>
<tr>
<td>FunctionProperties</td>
<td></td>
</tr>
<tr>
<td>DatabaseProperties</td>
<td></td>
</tr>
<tr>
<td>DeleteObject</td>
<td></td>
</tr>
<tr>
<td>DeleteVariable</td>
<td></td>
</tr>
<tr>
<td>DeleteFunction</td>
<td></td>
</tr>
<tr>
<td>DeleteDatabase</td>
<td></td>
</tr>
<tr>
<td>SelectVariable</td>
<td></td>
</tr>
<tr>
<td>SelectDatabase</td>
<td></td>
</tr>
<tr>
<td>NewVariableWiz</td>
<td></td>
</tr>
<tr>
<td>NewDatabaseWiz</td>
<td></td>
</tr>
</tbody>
</table>

PrinterName

Description:
PrinterName represents the name of the printer to which label will be printed.
Syntax:
PrinterName As String

Objects

Description:
Returns the interface to the object list.

Syntax:
Property Objects As IObjectList

Access Rights:
read-only

Variables

Description:
Returns the interface to the variable list for input variables.

Syntax:
Property Variables As IVariableList

Access Rights:
read-only

Functions

Description:
Returns the interface to the function list.

Syntax:
Property Functions As IFunctionList

Access Rights:
read-only

Databases

Description:
Returns the interface to the database list.
Syntax:
Property Databases As IDatabaseList

Access Rights:
read-only

**CurrUnit**

Description:
CurrUnit represent the current measurement unit used in the label. Possible values for are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Cm</td>
</tr>
<tr>
<td>1</td>
<td>Inch</td>
</tr>
<tr>
<td>2</td>
<td>Dot</td>
</tr>
<tr>
<td>3</td>
<td>Mm</td>
</tr>
</tbody>
</table>

Syntax:
Property CurrUnit As Long

Access Rights:
read-only

**GlobalVariables**

Description:
Returns the interface to the variable list for global variables.

Syntax:
Property GlobalVariables As IVariableList

Access Rights:
read-only

**NewVariable**

Description:
This method creates a new variable on the label.
Syntax:
Function NewVariable() As Boolean

NewFunction

Description:
This method creates a new function on the label.

Syntax:
Function NewFunction() As Boolean

NewDatabase

Description:
This method creates a new database access on the label.

Syntax:
Function NewDatabase() As Boolean

ObjectProperties

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

Syntax:
Function ObjectProperties(ID As Long) As Boolean

VariableProperties

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

Syntax:
Function VariableProperties(ID As Long) As Boolean

FunctionProperties

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

**Syntax:**
```
Function FunctionProperties[ID As Long] As Boolean
```

**DatabaseProperties**

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

**Syntax:**
```
Function DatabaseProperties[ID As Long] As Boolean
```

**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
```

**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
```

**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
```
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

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

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

NewVariableWiz

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

*Syntax:*
Function NewVariableWiz() As Boolean

NewDatabaseWiz

*Description:*
NewDatabaseWiz method opens Database Wizard. In case of success method returns value TRUE.
### Syntax:

Function NewDatabaseWiz() As Boolean

#### 3.3.14 Class ILabelGallery

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>LabelFileName</td>
</tr>
<tr>
<td>Save</td>
<td>PrinterName</td>
</tr>
<tr>
<td>Print</td>
<td></td>
</tr>
<tr>
<td>ExecuteMacro</td>
<td></td>
</tr>
<tr>
<td>ExportToPocket</td>
<td></td>
</tr>
<tr>
<td>SessionStart</td>
<td></td>
</tr>
<tr>
<td>SessionPrint</td>
<td></td>
</tr>
<tr>
<td>SessionEnd</td>
<td></td>
</tr>
<tr>
<td>GetLabelPreview</td>
<td></td>
</tr>
<tr>
<td>Objects</td>
<td></td>
</tr>
<tr>
<td>Variables</td>
<td></td>
</tr>
<tr>
<td>Functions</td>
<td></td>
</tr>
<tr>
<td>Databases</td>
<td></td>
</tr>
<tr>
<td>CurrUnit</td>
<td></td>
</tr>
<tr>
<td>GlobalVariables</td>
<td></td>
</tr>
<tr>
<td>NewVariable</td>
<td></td>
</tr>
<tr>
<td>NewFunction</td>
<td></td>
</tr>
<tr>
<td>NewDatabase</td>
<td></td>
</tr>
<tr>
<td>ObjectProperties</td>
<td></td>
</tr>
<tr>
<td>VariableProperties</td>
<td></td>
</tr>
<tr>
<td>FunctionProperties</td>
<td></td>
</tr>
<tr>
<td>DatabaseProperties</td>
<td></td>
</tr>
<tr>
<td>DeleteObject</td>
<td></td>
</tr>
<tr>
<td>DeleteVariable</td>
<td></td>
</tr>
<tr>
<td>DeleteFunction</td>
<td></td>
</tr>
<tr>
<td>DeleteDatabase</td>
<td></td>
</tr>
<tr>
<td>SelectVariable</td>
<td></td>
</tr>
<tr>
<td>SelectDatabase</td>
<td></td>
</tr>
<tr>
<td>NewVariableWiz</td>
<td></td>
</tr>
<tr>
<td>NewDatabaseWiz</td>
<td></td>
</tr>
</tbody>
</table>
**LabelFileName**

*Description:*
LabelFileName represents the name of the label file.

*Syntax:*
LabelFileName As String

**PrinterName**

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

*Syntax:*
PrinterName As String

**Application**

*Description:*
Application method returns an interface to the GalleryApp.

*Syntax:*
Property Application As LabelGallery

*Access Rights:*
read-only

**Save**

*Description:*
Save method saves the selected label.

*Syntax:*
Function Save() As Boolean

**Print**

*Description:*
Print method prints as many labels as selected in Quantity. Labels are printed to the printer which is selected in PrinterName property.
**Syntax:**
Function Print(Quantity As String) As Boolean

**ExecuteMacro**

**Description:**
This function executes a macro. Macro parameter represents GalleryCommand.

**Syntax:**
Function ExecuteMacro(Macro As String) As Boolean

**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 GalleryCommands and it is not necessary to generate a complete data stream for each print session.

**Syntax:**
Function SessionStart() As Boolean

**See also:**
SessionPrint
SessionEnd

**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

LabelSessionEnd

Description:
The function closes data stream.

Syntax:
Function SessionEnd() As Boolean

See also:
LabelSessionStart
LabelSessionPrint

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 successfully return value is TRUE.

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

Objects

Description:
Returns the interface to the object list.

Syntax:
Property Objects As IObjectList
**Access Rights:**
read-only

**Variables**

**Description:**
Returns the interface to the variable list for input variables.

**Syntax:**
Property Variables As IVariableList

**Access Rights:**
read-only

**Functions**

**Description:**
Returns the interface to the function list.

**Syntax:**
Property Functions As IFunctionList

**Access Rights:**
read-only

**Databases**

**Description:**
Returns the interface to the database list.

**Syntax:**
Property Databases As IDatabaseList

**Access Rights:**
read-only

**CurrUnit**

**Description:**
CurrUnit represent the current measurement unit used in the label.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Cm</td>
</tr>
<tr>
<td>1</td>
<td>Inch</td>
</tr>
<tr>
<td>2</td>
<td>Dot</td>
</tr>
<tr>
<td>3</td>
<td>Mm</td>
</tr>
</tbody>
</table>

**Syntax:**

Property CurrUnit As Long

**Access Rights:**

read-only

**GlobalVariables**

**Description:**

Returns the interface to the variable list for global variables.

**Syntax:**

Property GlobalVariables As IVariableList

**Access Rights:**

read-only

**NewVariable**

**Description:**

This method creates a new variable on the label.

**Syntax:**

Function NewVariable() As Boolean

**NewFunction**

**Description:**
This method creates a new function on the label.

**Syntax:**
Function NewFunction() As Boolean

---

**NewDatabase**

**Description:**
This method creates a new database access on the label.

**Syntax:**
Function NewDatabase() As Boolean

---

**ObjectProperties**

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

**Syntax:**
Function ObjectProperties(ID As Long) As Boolean

---

**VariableProperties**

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

**Syntax:**
Function VariableProperties(ID As Long) As Boolean

---

**FunctionProperties**

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

**Syntax:**
Function FunctionProperties(ID As Long) As Boolean
**DatabaseProperties**

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

*Syntax:*
Function DatabaseProperties(ID As Long) As Boolean

**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

**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

**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

**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

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

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

NewVariableWiz

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

Syntax:
Function NewVariableWiz() As Boolean

NewDatabaseWiz

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

Syntax:
Function NewDatabaseWiz() As Boolean
3.3.15 Class IObject

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Height</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>ZOrder</td>
</tr>
<tr>
<td>Variable</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td></td>
</tr>
<tr>
<td>SetVariable</td>
<td></td>
</tr>
</tbody>
</table>

**Name**

**Description:**
Name property represents the name of the object.

**Syntax:**
Name As String

**Selected**

**Description:**
When the element is selected, this property is TRUE.

**Syntax:**
Selected As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

**Syntax:**

Rotation As Long

**IsLocked**

**Description:**

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

**Syntax:**

IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
<tr>
<td>8</td>
<td>bottom right</td>
</tr>
</tbody>
</table>

Syntax:
AnchorPoint As Long

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:

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>
Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

**Syntax:**

FormatID As Long

**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**

*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

**ID**

*Description:*
ID of the element

**Syntax:**

Property ID 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

---

**Top**

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

**Syntax:**
Property Top As Long

**Access Rights:**
read-only

---

**Width**

**Description:**
Width of the element (in 0.01 mm units).

**Syntax:**
Property Width As Long

**Access Rights:**
read-only

---

**Height**

**Description:**
Height of the element (in 0.01 mm units).

**Syntax:**
Property Height As Long

**Access Rights:**
read-only
Kind

*Description:*
Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
<tr>
<td>307</td>
<td>OleObject</td>
</tr>
<tr>
<td>308</td>
<td>Downloaded Graphic Object</td>
</tr>
<tr>
<td>309</td>
<td>ParagraphObject</td>
</tr>
<tr>
<td>312</td>
<td>RTFTextObject</td>
</tr>
<tr>
<td>313</td>
<td>EllipseObject</td>
</tr>
</tbody>
</table>

*Syntax:*

Property Kind As Long

*Access Rights:*
read-only

Status

*Description:*
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>
**Syntax:**
Property Status As Long

**Access Rights:**
read-only

**RotateFlag**

**Description:**
Flag, which defines, how the object can be rotated.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**
Property RotateFlag As Long

**Access Rights:**
read-only

**ResizeFlag**

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

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

**Syntax:**
Property ResizeFlag As Long
Access Rights:
read-only

**Variable**

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

*Syntax:*
Property Variable As IVar

**Move**

*Description:*
Move the element to the location X, Y

*Syntax:*
Move(X As Long, Y 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)

**SetVariable**

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

*Syntax:*
Function SetVariable(ID As Long) As Long

3.3.16 Class IObjectList

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
</tbody>
</table>

**Count**

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

*Syntax:*
Property Count As Long

*Access Rights:*
read-only

**Item**

*Description:*
Returns the interface to the object. Object is selected with Index.

*Syntax:*
Function Item(Index As Long) As IObject

3.3.17 Class IOleObject

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Height</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>ZOrder</td>
</tr>
<tr>
<td>------------</td>
<td>--------</td>
</tr>
<tr>
<td>Variable</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Move</td>
<td></td>
</tr>
<tr>
<td>Resize</td>
<td></td>
</tr>
<tr>
<td>SetVariable</td>
<td></td>
</tr>
</tbody>
</table>

**Name**

*Description:*
Name property represents the name of the OLE object.

*Syntax:*
Name As String

**Selected**

*Description:*
When the element is selected, this property is TRUE.

*Syntax:*
Selected As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

When the element can be rotated in steps of 1 degree, the property has the value from 0 to 359.
**Syntax:**
Rotation As Long

**IsLocked**

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

**Syntax:**
IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
</tbody>
</table>
### Syntax:

AnchorPoint As Long

### 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

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

**Syntax:**

FormatID As Long

### 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**

**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

**ID**

**Description:**
ID of the element

**Syntax:**
Property ID 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

**Top**

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

**Syntax:**
Property Top As Long

Access Rights:
read-only

**Width**

Description:
Width of the element (in 0.01 mm units).

Syntax:
Property Width As Long

Access Rights:
read-only

**Height**

Description:
Height of the element (in 0.01 mm units).

Syntax:
Property Height As Long

Access Rights:
read-only

**Kind**

Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
</tbody>
</table>
ActiveX and Dynamic Data Exchange Programmers Interface

### Syntax:
```
Property Kind As Long
```

**Access Rights:**
read-only

### Status

**Description:**
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

**Syntax:**
```
Property Status As Long
```

**Access Rights:**
read-only

### RotateFlag

**Description:**
Flag, which defines, how the object can be rotated.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
</tbody>
</table>
# ActiveX and Dynamic Data Exchange Programmers Interface

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**

Property RotateFlag As Long

**Access Rights:**

read-only

## ResizeFlag

**Description:**

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

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

**Syntax:**

Property ResizeFlag As Long

**Access Rights:**

read-only

## Variable

**Description:**

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

**Syntax:**

Property Variable As IVar

**Access Rights:**

read-only
Move

Description:
Move the element to the location X, Y

Syntax:
Move(X As Long, Y 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)

SetVariable

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

Syntax:
Function SetVariable(ID As Long) As Long

3.3.18 Class IParagraph

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Height</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>ZOrder</td>
</tr>
<tr>
<td>Variable</td>
<td>PageNumber</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>Move</td>
<td>FontName</td>
</tr>
<tr>
<td>Resize</td>
<td>IsItalic</td>
</tr>
<tr>
<td>SetVariable</td>
<td>IsBold</td>
</tr>
<tr>
<td>Contents</td>
<td>IsInverse</td>
</tr>
<tr>
<td>SetContents</td>
<td>IsMirror</td>
</tr>
<tr>
<td>PtSizeX</td>
<td></td>
</tr>
<tr>
<td>PtSizeY</td>
<td></td>
</tr>
<tr>
<td>SpacingX</td>
<td></td>
</tr>
<tr>
<td>SpacingY</td>
<td></td>
</tr>
<tr>
<td>BestFit</td>
<td></td>
</tr>
<tr>
<td>MinSizeX</td>
<td></td>
</tr>
<tr>
<td>MinSizeY</td>
<td></td>
</tr>
<tr>
<td>MaxSizeX</td>
<td></td>
</tr>
<tr>
<td>MaxSizeY</td>
<td></td>
</tr>
</tbody>
</table>

**Name**

**Description:**
Name property represents the name of the paragraph..

**Syntax:**
Name As String

**Selected**

**Description:**
When the element is selected, this property is TRUE.

**Syntax:**
Selected As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

**Syntax:**
Rotation As Long

**IsLocked**

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

**Syntax:**
IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
<tr>
<td>8</td>
<td>bottom right</td>
</tr>
</tbody>
</table>

Syntax:
AnchorPoint As Long

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>
Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.

**Syntax:**

FormatID As Long

---

**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**

**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

---

**FontName**

**Description:**

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

**Syntax:**

FontName As String

---

**IsItalic**

**Description:**

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

**Syntax:**
**IsItalic**

**As Boolean**

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

**Syntax:**
IsItalic As Boolean

---

**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

---

**IsMirror**

**Description:**
This property is TRUE if Mirror efect is enabled.

**Syntax:**
IsMirror As Boolean

---

**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

---

**PtSizeY**

**Description:**
PtSizeY represents vertical font size.
Syntax:
PtSizeY As Long

**SpacingX**

*Description:*
With SpacingX property you select spacing between each character.

*Syntax:*
SpacingX As Long

**SpacingY**

*Description:*
With SpacingY property you select spacing between each line.

*Syntax:*
SpacingY As Long

**BestFit**

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

*Syntax:*
BestFit As Boolean

**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

**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

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

**Syntax:**
MaxSizeX As Long

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

**Syntax:**
MaxSizeY As Long

**ID**

**Description:**
ID of the element

**Syntax:**
Property ID 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
**Top**

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

*Syntax:*
Property Top As Long

*Access Rights:*
read-only

**Width**

*Description:*
Width of the element (in 0.01 mm units).

*Syntax:*
Property Width As Long

*Access Rights:*
read-only

**Height**

*Description:*
Height of the element (in 0.01 mm units).

*Syntax:*
Property Height As Long

*Access Rights:*
read-only

**Kind**
Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------</td>
</tr>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

### Syntax:
```
Property Kind As Long
```

### Access Rights:
read-only

### Status

#### Description:
Status of the object.

Possible values are:

### Syntax:
```
Property Status As Long
```

### Access Rights:
read-only

### RotateFlag

#### Description:
Flag, which defines, how the object can be rotated.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**
Property RotateFlag As Long

**Access Rights:**
read-only

**ResizeFlag**
Description:
Flag, which defines, how the object can be resized.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

Syntax:
Property ResizeFlag As Long

Access Rights:
read-only

**Variable**

**Description:**
Returns the interface to the variable, which is connected to the paragraph.
Syntax:
Property Variable As IVar

Access Rights:
read-only

**Move**

Description:
Move the element to the location X, Y

Syntax:
Move(X As Long, Y 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)

**SetVariable**

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

Syntax:
Function SetVariable(ID As Long) As Long

**Contents**

Description:
Returns the current contents of the element.

Syntax:
Property Contents As String
Access Rights:
read-only

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

3.3.19 Class IParameterList

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td></td>
</tr>
</tbody>
</table>

Count

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

Syntax:
Property Count As Long

Access Rights:
read-only

Item

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

Syntax:
Function Item(Index As Long) As IDbParameter
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. Output is an interface to the new created filter.

Syntax:
Function Create(Field As String) As IDBParameter

3.3.20 Class IRectangle

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Height</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>ZOrder</td>
</tr>
<tr>
<td>Move</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Resize</td>
<td>ThicknessX</td>
</tr>
<tr>
<td></td>
<td>ThicknessY</td>
</tr>
<tr>
<td></td>
<td>Radius</td>
</tr>
<tr>
<td></td>
<td>LineStyle</td>
</tr>
<tr>
<td></td>
<td>FillStyle</td>
</tr>
<tr>
<td></td>
<td>IsRounded</td>
</tr>
<tr>
<td></td>
<td>PrintAsGraphics</td>
</tr>
</tbody>
</table>

Name

Description:
Name property represents the name of the rectangle.

Syntax:
Name As String
Selected

_Description:
When the element is selected, this property is TRUE.

_Syntax:
Selected As Boolean

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

_Syntax:
Rotation As Long

IsLocked

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

_Syntax:
IsLocked As Boolean

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
<tr>
<td>8</td>
<td>bottom right</td>
</tr>
</tbody>
</table>

**Syntax:**
AnchorPoint As Long

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

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

**Syntax:**

FormatID As Long

**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**

**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
**ThicknessX**

ThicknessX As Long

**ThicknessY**

*Description:*
ThicknessY property represents the thickness of the vertical lines of the rectangle.

*Syntax:*
ThicknessY As Long

**Radius**

*Description:*
Radius property represents the radius of the rectangle corners.

*Syntax:*
Radius As Long

**LineStyle**

*Description:*
LineStyle property represents the line style for the rectangle.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Solid</td>
</tr>
<tr>
<td>2</td>
<td>Dash</td>
</tr>
<tr>
<td>4</td>
<td>Dot</td>
</tr>
<tr>
<td>8</td>
<td>Dash + Dot</td>
</tr>
<tr>
<td>16</td>
<td>Dash + Dot + Dot</td>
</tr>
<tr>
<td>32</td>
<td>Clear</td>
</tr>
</tbody>
</table>

*Syntax:*
LineStyle As Long
**FillStyle**

*Description:*
FillStyle property represents the fill style for the rectangle. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Clear</td>
</tr>
<tr>
<td>2</td>
<td>Solid</td>
</tr>
<tr>
<td>3</td>
<td>Right diagonal</td>
</tr>
<tr>
<td>4</td>
<td>Left diagonal</td>
</tr>
<tr>
<td>8</td>
<td>Vertical</td>
</tr>
<tr>
<td>16</td>
<td>Horizontal</td>
</tr>
<tr>
<td>32</td>
<td>Cross</td>
</tr>
<tr>
<td>64</td>
<td>Cross diagonal</td>
</tr>
<tr>
<td>128</td>
<td>25 % of color</td>
</tr>
<tr>
<td>256</td>
<td>50 % of color</td>
</tr>
<tr>
<td>512</td>
<td>75 % of color</td>
</tr>
</tbody>
</table>

*Syntax:*
FillStyle As Long

**IsRounded**

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

*Syntax:*
IsRounded As Boolean

**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

ID

Description:
ID of the element

Syntax:
Property ID 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

Top

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

Syntax:
Property Top As Long

Access Rights:
read-only

Width

Description:
Width of the element (in 0.01 mm units).
**Syntax:**
Property Width As Long

**Access Rights:**
read-only

### Height

**Description:**
Height of the element (in 0.01 mm units).

**Syntax:**
Property Height As Long

**Access Rights:**
read-only

Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
<tr>
<td>307</td>
<td>OleObject</td>
</tr>
<tr>
<td>308</td>
<td>Downloaded Graphic Object</td>
</tr>
<tr>
<td>309</td>
<td>ParagraphObject</td>
</tr>
<tr>
<td>312</td>
<td>RTFTextObject</td>
</tr>
<tr>
<td>313</td>
<td>EllipseObject</td>
</tr>
</tbody>
</table>

**Syntax:**
Property Kind As Long

**Access Rights:**
read-only
**Status**

*Description:*
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

*Syntax:*

Property Status As Long

*Access Rights:*
read-only

**RotateFlag**

*Description:*
Flag, which defines, how the object can be rotated.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

*Syntax:*

Property RotateFlag As Long

*Access Rights:*
read-only

**ResizeFlag**

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

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

Syntax:
Property ResizeFlag As Long

Access Rights:
read-only

**Move**

**Description:**
Move the element to the location X, Y

**Syntax:**
Move(X As Long, Y 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)

### 3.3.21 Class IRTFText

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
</tbody>
</table>
### Name

**Description:**
Name property represents the name of the RTF text.

**Syntax:**
Name As String

### Selected

**Description:**
When the element is selected, this property is TRUE.

**Syntax:**
Selected As Boolean

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

**Syntax:**
Rotation As Long

**IsLocked**

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

**Syntax:**
IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
<tr>
<td>5</td>
<td>mid right</td>
</tr>
<tr>
<td>6</td>
<td>bottom left</td>
</tr>
<tr>
<td>7</td>
<td>bottom center</td>
</tr>
<tr>
<td>8</td>
<td>bottom right</td>
</tr>
</tbody>
</table>

**Syntax:**

AnchorPoint As Long

---

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

Details about the characters in each format can be found in formats.def file, located on BIN\SYSTEM directory.
**Syntax:**
FormatID As Long

**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**

**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

**ID**

**Description:**
ID of the element

**Syntax:**
Property ID 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

**Top**

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

Syntax:
Property Top As Long

Access Rights: read-only

**Width**

Description:
Width of the element (in 0.01 mm units).

Syntax:
Property Width As Long

Access Rights: read-only

**Height**

Description:
Height of the element (in 0.01 mm units).

Syntax:
Property Height As Long

Access Rights: read-only

**Kind**
Element kind.

Possible values are:
### Value and Description

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
<tr>
<td>307</td>
<td>OleObject</td>
</tr>
<tr>
<td>308</td>
<td>Downloaded Graphic Object</td>
</tr>
<tr>
<td>309</td>
<td>ParagraphObject</td>
</tr>
<tr>
<td>312</td>
<td>RTFTextObject</td>
</tr>
<tr>
<td>313</td>
<td>EllipseObject</td>
</tr>
</tbody>
</table>

**Syntax:**
```
Property Kind As Long
```

**Access Rights:**
read-only

### Status

**Description:**
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

**Syntax:**
```
Property Status As Long
```

**Access Rights:**
**RotateFlag**

*Description:*
Flag, which defines, how the object can be rotated.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

*Syntax:*
Property RotateFlag As Long

*Access Rights:*
read-only

**ResizeFlag**

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

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

Syntax:
Property ResizeFlag As Long

Access Rights:
read-only
Move

**Description:**
Move the element to the location X, Y

**Syntax:**
Move(X As Long, Y 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)

### 3.3.22 Class IText

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>Left</td>
<td>Selected</td>
</tr>
<tr>
<td>Top</td>
<td>Rotation</td>
</tr>
<tr>
<td>Width</td>
<td>IsLocked</td>
</tr>
<tr>
<td>Height</td>
<td>AnchorPoint</td>
</tr>
<tr>
<td>Kind</td>
<td>AnchorElementID</td>
</tr>
<tr>
<td>Status</td>
<td>AnchorLevel</td>
</tr>
<tr>
<td>RotateFlag</td>
<td>FormatID</td>
</tr>
<tr>
<td>ResizeFlag</td>
<td>ZOrder</td>
</tr>
<tr>
<td>Variable</td>
<td>PageNumber</td>
</tr>
<tr>
<td>Move</td>
<td>FontName</td>
</tr>
<tr>
<td>Resize</td>
<td>IsItalic</td>
</tr>
<tr>
<td>SetVariable</td>
<td>IsBold</td>
</tr>
<tr>
<td>Contents</td>
<td>IsInverse</td>
</tr>
<tr>
<td>SetContents</td>
<td>IsMirror</td>
</tr>
<tr>
<td>SetContents</td>
<td>PtSizeX</td>
</tr>
<tr>
<td></td>
<td>PtSizeY</td>
</tr>
<tr>
<td></td>
<td>SpacingX</td>
</tr>
</tbody>
</table>
**Name**

*Description:*
Name property represents the name of the RTF text.

*Syntax:*
Name As String

**Selected**

*Description:*
When the element is selected, this property is TRUE.

*Syntax:*
Selected As Boolean

**Rotation**

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

Possible values
The valid values for the property are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 deg.</td>
</tr>
<tr>
<td>1</td>
<td>90 deg.</td>
</tr>
<tr>
<td>2</td>
<td>180 deg.</td>
</tr>
<tr>
<td>3</td>
<td>270 deg.</td>
</tr>
</tbody>
</table>

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

*Syntax:*
Rotation As Long
**IsLocked**

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

*Syntax:*
IsLocked As Boolean

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>top left</td>
</tr>
<tr>
<td>1</td>
<td>top center</td>
</tr>
<tr>
<td>2</td>
<td>top right</td>
</tr>
<tr>
<td>3</td>
<td>mid left</td>
</tr>
<tr>
<td>4</td>
<td>mid center</td>
</tr>
</tbody>
</table>
### Syntax:
AnchorPoint As Long

### 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

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

**Syntax:**
FormatID As Long

### 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**

*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

**FontName**

*Description:*
FontName property represents the name of the font used within paragraph.

*Syntax:*
FontName As String

**IsItalic**

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

*Syntax:*
IsItalic As Boolean

**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 effect is enabled.
**Syntax:**
IsInverse As Boolean

**IsMirror**

*Description:*
This property is TRUE if Mirror effect is enabled.

*Syntax:*
IsMirror As Boolean

**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

**PtSizeY**

*Description:*
PtSizeY represents vertical font size.

*Syntax:*
PtSizeY As Long

**SpacingX**

*Description:*
With SpacingX property you select spacing between each character.

*Syntax:*
SpacingX As Long

**SpacingY**

*Description:*

With SpacingY property you select spacing between each line.

**Syntax:**
SpacingY As Long

**ID**

**Description:**
ID of the element

**Syntax:**
Property ID 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

**Top**

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

**Syntax:**
Property Top As Long

**Access Rights:**
read-only
Width

**Description:**
Width of the element (in 0.01 mm units).

**Syntax:**
Property Width As Long

**Access Rights:**
read-only

Height

**Description:**
Height of the element (in 0.01 mm units).

**Syntax:**
Property Height As Long

**Access Rights:**
read-only

Kind

Element kind.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>301</td>
<td>TextObject</td>
</tr>
<tr>
<td>302</td>
<td>RectangleObject</td>
</tr>
<tr>
<td>303</td>
<td>BitmapObject</td>
</tr>
<tr>
<td>304</td>
<td>BarcodeObject</td>
</tr>
<tr>
<td>305</td>
<td>LineObject</td>
</tr>
<tr>
<td>306</td>
<td>InverseObject</td>
</tr>
<tr>
<td>307</td>
<td>OleObject</td>
</tr>
<tr>
<td>308</td>
<td>Downloaded Graphic Object</td>
</tr>
<tr>
<td>309</td>
<td>ParagraphObject</td>
</tr>
<tr>
<td>312</td>
<td>RTFTextObject</td>
</tr>
<tr>
<td>313</td>
<td>EllipseObject</td>
</tr>
</tbody>
</table>
**Syntax:**
Property Kind As Long

**Access Rights:**
read-only

**Status**

**Description:**
Status of the object.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OK</td>
</tr>
<tr>
<td>1</td>
<td>Error condition</td>
</tr>
<tr>
<td>2</td>
<td>Phantom</td>
</tr>
</tbody>
</table>

**Syntax:**
Property Status As Long

**Access Rights:**
read-only

**RotateFlag**

**Description:**
Flag, which defines, how the object can be rotated.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no rotation</td>
</tr>
<tr>
<td>0x07</td>
<td>90 degrees rotation</td>
</tr>
<tr>
<td>0x02</td>
<td>180 degrees rotation</td>
</tr>
<tr>
<td>0xFF</td>
<td>0-360 degrees rotation in steps of 1 degree</td>
</tr>
</tbody>
</table>

**Syntax:**
Property RotateFlag As Long
Access Rights:
read-only

ResizeFlag
Description:
Flag, which defines, how the object can be resized.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>no resizing</td>
</tr>
<tr>
<td>0x88</td>
<td>resizing is possible only in horizontal direction</td>
</tr>
<tr>
<td>0x22</td>
<td>resizing is possible only in vertical direction</td>
</tr>
<tr>
<td>0x55</td>
<td>resizing is possible only in both directions at the same time</td>
</tr>
</tbody>
</table>

Syntax:
Property ResizeFlag As Long

Access Rights:
read-only

Variable

Description:
Returns the interface to the variable, which is connected to the text.

Syntax:
Property Variable As IVar

Access Rights:
read-only

Move

Description:
Move the element to the location X, Y

Syntax:
Move(X As Long, Y 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)

**SetVariable**

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

*Syntax:*
Function SetVariable(ID As Long) As Long

**Contents**

*Description:*
Returns the current contents of the element.

*Syntax:*
Property Contents As String

*Access Rights:*
read-only

**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`

### 3.3.23 Class IVar

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Name</td>
</tr>
<tr>
<td>VarType</td>
<td>Description</td>
</tr>
<tr>
<td>IncrementType</td>
<td>Prefix</td>
</tr>
<tr>
<td>IncrementKind</td>
<td>Suffix</td>
</tr>
<tr>
<td>IsUsed</td>
<td>JustificationType</td>
</tr>
<tr>
<td>Default</td>
<td>PadCharacter</td>
</tr>
<tr>
<td>SetValue</td>
<td>FixecLength</td>
</tr>
<tr>
<td>GetValue</td>
<td>Length</td>
</tr>
<tr>
<td>DependsOnFunction</td>
<td>IsMultiLine</td>
</tr>
<tr>
<td>ShouldTraceNow</td>
<td>WordWrap</td>
</tr>
<tr>
<td></td>
<td>BreakLines</td>
</tr>
<tr>
<td></td>
<td>LineLength</td>
</tr>
<tr>
<td></td>
<td>LineCount</td>
</tr>
<tr>
<td></td>
<td>IsVarQuantity</td>
</tr>
<tr>
<td></td>
<td>StrictChecking</td>
</tr>
<tr>
<td></td>
<td>HasMinValue</td>
</tr>
<tr>
<td></td>
<td>MinValue</td>
</tr>
<tr>
<td></td>
<td>HasMaxValue</td>
</tr>
<tr>
<td></td>
<td>MaxValue</td>
</tr>
<tr>
<td></td>
<td>FormatID</td>
</tr>
<tr>
<td></td>
<td>IsBasedOnQuantity</td>
</tr>
<tr>
<td></td>
<td>PromptCount</td>
</tr>
<tr>
<td></td>
<td>Prompt</td>
</tr>
<tr>
<td></td>
<td>DefaultValue</td>
</tr>
<tr>
<td></td>
<td>DefType</td>
</tr>
<tr>
<td></td>
<td>DynamicValue</td>
</tr>
<tr>
<td></td>
<td>ValueRequired</td>
</tr>
<tr>
<td></td>
<td>IncrementCount</td>
</tr>
<tr>
<td></td>
<td>IncrementStep</td>
</tr>
<tr>
<td></td>
<td>IncrementStep2</td>
</tr>
<tr>
<td></td>
<td>IncrementStep3</td>
</tr>
<tr>
<td></td>
<td>RollOver</td>
</tr>
</tbody>
</table>
Name

Description:
Name property represents the name of the variable. Name can be only 12 characters long.

Syntax:
Name As String

Description

Description:
Description property represents the description of the variable.

Syntax:
Description As String

Prefix

Description:
Prefix property represents the prefix of the variable.

Syntax:
Prefix As String

Suffix

Description:
Suffix property represents the suffix of the variable.

Syntax:
Suffix As String

JustificationType
JustificationType property represents how the contents of the variable will be justified. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Right</td>
</tr>
<tr>
<td>2</td>
<td>Left</td>
</tr>
<tr>
<td>3</td>
<td>Center</td>
</tr>
<tr>
<td>4</td>
<td>Full</td>
</tr>
</tbody>
</table>

For the variables full justification type is not used.

**Syntax:**

JustificationType As Long

**PadCharacter**

**Description:**
PadCharacter property represents the PadCharacter of the variable.

**Syntax:**

PadCharacter As String

**FixedLength**

**Description:**
If variable has fixed length then FixedLength property has value TRUE.

**Syntax:**

FixedLength As Boolean

**Length**

**Description:**
Length property represents the length of the variable. Length can have value between 1 and 4096.

**Syntax:**

Length As Long
**IsMultiLine**

*Description:*
If variable has MultiLine option enabled then value of the IsMultiLine property is TRUE.

*Syntax:*
IsMultiLine As Boolean

**WordWrap**

*Description:*
If variable has WordWrap option enabled then value of the WordWrap property is TRUE.

*Syntax:*
WordWrap As Boolean

**BreakLines**

*Description:*
If variable has BreakLines option enabled then value of the WordWrap property is TRUE.

*Syntax:*
BreakLines As Boolean

**LineLength**

*Description:*
LineLength property represents the length of the one line. LineLength can have value between 1 and 4096.

*Syntax:*
LineLength As Long

**LineCount**

*Description:*
LineCount property represents the number of lines in one variable. LineCount can have value between 1 and 100.

*Syntax:*

LineCount As Long

**IsVarQuantity**

*Description:*
IsVarQuantity property has value TRUE, if "Treat as variable quantity" is enabled.

*Syntax:*
IsVarQuantity 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

**HasMinValue**

*Description:*
HasMinValue property has value TRUE, if range check option Min value is enabled.

*Syntax:*
HasMinValue As Boolean

**MinValue**

*Description:*
MinValue represents the minimal value for range checking.

*Syntax:*
MinValue As String

**HasMaxValue**

*Description:*
HasMaxValue property has value TRUE, if range check option Max value is enabled.
Syntax:
HasMaxValue As Boolean

MaxValue

Description:
MaxValue represents the minimal value for range checking.

Syntax:
MaxValue 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All</td>
</tr>
<tr>
<td>1</td>
<td>Numeric</td>
</tr>
<tr>
<td>2</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>3</td>
<td>Letters</td>
</tr>
<tr>
<td>4</td>
<td>7 bit</td>
</tr>
<tr>
<td>5</td>
<td>Hex</td>
</tr>
<tr>
<td>6</td>
<td>Date</td>
</tr>
<tr>
<td>7</td>
<td>Time</td>
</tr>
</tbody>
</table>

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

Syntax:
FormatID As Long

IsBasedOnQuantity

Description:
IsBasedOnQuantity property has value TRUE, if "Based on var. quantity" is enabled.
**Syntax:**
IsBasedOnQuantity As Boolean

**PromptCount**

**Description:**
PromptCount property represents the number on how many printed labels it will be prompted.

**Syntax:**
PromptCount As Long

**Prompt**

**Description:**
In Prompt property is stored a prompt string.

**Syntax:**
Prompt As String

**DefaultValue**

**Description:**
In DefaultValue property is stored a default value for the variable.

**Syntax:**
DefaultValue As String

**DefType**

**Description:**
DefType property represents the case on which default value for the variable will be selected.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None – No default value</td>
</tr>
<tr>
<td>1</td>
<td>Prompt – You are prompted for default value</td>
</tr>
</tbody>
</table>
2 No prompt – Default value is stored in Default value property

**Syntax:**

DefType As Long

**DynamicValue**

**Description:**

DynamicValue has value TRUE, if "Dynamic value" option for default value is enabled.

**Syntax:**

DynamicValue As Boolean

**ValueRequired**

**Description:**

ValueRequired has value TRUE, if "ValueRequired" option for prompted variable is enabled.

**Syntax:**

ValueRequired As Boolean

**IncrementCount**

**Description:**

IncrementCount property represents the number on how many printed labels counter will be incremented.

**Syntax:**

IncrementCount As Long

**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

IncrementStep2

Description:
See IncrementStep

Syntax:
IncrementStep2 As Long

IncrementStep3

Description:
See IncrementStep

Syntax:
IncrementStep3 As Long

RollOver

Description:
RollOver has value TRUE, if "RollOver" option is enabled.

Syntax:
RollOver As Boolean
**TraceOn**

*Description:*  
TraceOn has value TRUE, if variable tracing is enabled.

*Syntax:*  
TraceOn As Boolean

**PictureType**

*Description:*  
Picture type property represents, which picture for the variable is selected.
Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Date</td>
</tr>
<tr>
<td>2</td>
<td>Time</td>
</tr>
<tr>
<td>3</td>
<td>Floating point</td>
</tr>
<tr>
<td>4</td>
<td>Money</td>
</tr>
<tr>
<td>5</td>
<td>List</td>
</tr>
<tr>
<td>6</td>
<td>Binary</td>
</tr>
</tbody>
</table>

*Syntax:*  
PictureType As Long

**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 possibilities for input pictures.

*Syntax:*  
InputPicture 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 possibilities for output pictures.

**Syntax:**
OutputPicture As String

**ID**

**Description:**
ID of the element

**Syntax:**
Property ID As Long

**Access Rights:**
read-only

**VarType**

**Description:**
Vartype property represents the type of the variable.

Vartype property can have following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prompt</td>
</tr>
<tr>
<td>2</td>
<td>System clock</td>
</tr>
<tr>
<td>3</td>
<td>Printer clock</td>
</tr>
<tr>
<td>4</td>
<td>Global</td>
</tr>
<tr>
<td>5</td>
<td>Generated</td>
</tr>
<tr>
<td>6</td>
<td>Overload</td>
</tr>
<tr>
<td>7</td>
<td>Database</td>
</tr>
<tr>
<td>8</td>
<td>Contents provider</td>
</tr>
</tbody>
</table>

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

**IncrementType**

**Description:**

IncrementType property represents the type of incrementation.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>Increment</td>
</tr>
<tr>
<td>2</td>
<td>Decrement</td>
</tr>
<tr>
<td>3</td>
<td>Offset</td>
</tr>
</tbody>
</table>

**Syntax:**

Property IncrementType As Long

**Access Rights:**

read-only

**IncrementKind**

**Description:**

IncrementKind property represents the kind of the increment. Increment can be performed on several types of variables.

Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Integer</td>
</tr>
</tbody>
</table>
Syntax:
Property IncrementKind As Long

Access Rights:
read-only

**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

**Default**

Description:
Default method returns the current value of the variable.

Syntax:
Property Default As String

Access Rights:
read-only

**SetValue**

Description:
SetValue method is used for setting the value for the variable.

Syntax:
Function SetValue(Value As String) As Long
**GetValue**

*Description:*
GetValue method returns the current value of the variable.

*Syntax:*
Function GetValue() As String

**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

**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

### 3.3.24 Class IVariableList

<table>
<thead>
<tr>
<th>Methods</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>&lt;none&gt;</td>
</tr>
<tr>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>FindByName</td>
<td></td>
</tr>
<tr>
<td>FindByID</td>
<td></td>
</tr>
<tr>
<td>Create</td>
<td></td>
</tr>
<tr>
<td>CreatePrompted</td>
<td></td>
</tr>
<tr>
<td>Add</td>
<td></td>
</tr>
<tr>
<td>Remove</td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td></td>
</tr>
<tr>
<td>FlushList</td>
<td></td>
</tr>
</tbody>
</table>
Count

Description:
Count property returns the number of variables, which are defined on the label.

Syntax:
Property Count As Long

Access Rights:
read-only

Item

Description:
Returns the interface to the existing variable. Variable is selected with Index.

Syntax:
Function Item(Index As Long) As IVar

FindByNamed

Description:

Description:
Returns the interface to the existing variable. Variable is selected with Name.

Syntax:
Function FindByName(Name As String) As IVar

FindByID

Description:

Description:
Returns the interface to the existing variable. Variable is selected with ID.

Syntax:
Function FindByID(ID As Long) As IVar

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

**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

**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

**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

**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
FlushList

Description:
FlushList method erases the whole variable list. If the operation was successful TRUE value is returned.

Syntax:
Function FlushList() As Boolean
4. Programming Samples

4.1 DDE

DDE programming samples for Microsoft Excel, Microsoft Word and Microsoft Access are installed together with the installation of LabelGallery. Location for the DDE programming samples is "..\Program Files\Sato\Gallery\Samples\Integration"

4.2 ActiveX interface

ActiveX programming samples for Visual Basic and Borland Delphi are installed together with the installation of LabelGallery. Location for the Active programming samples is "..\Sato\Gallery\Samples\Integration"

4.2.1 ActiveX Hints

Here is described how to speed up printing with LabelGallery thru ActiveX interface. Following you will find four options how communication can be performed. They are sorted from the slowest to the fastest one.

VERY VERY SLOW:

Description:
The main reason this option is the slowest one is that every time when label is going to be opened LabelGallery object is created. After that every label is opened and closed for every printout.

Syntax:
Set GalleryApp = createObject ("LabelGalleryPlus.Application")
LabelID = GalleryApp.LabelOpen("sample3.lbl")
Success = GalleryApp.LabelSetVar(LabelID,"Bar Code","11",0,0)
Success = GalleryApp.LabelPrint(LabelID,1)    ' Print 1 label
Success = GalleryApp.LabelClose(LabelID)
GalleryApp.Quit

Set GalleryApp = createObject ("LabelGalleryPlus.Application")
LabelID = GalleryApp.LabelOpen("sample3.lbl")
Success = GalleryApp.LabelSetVar(LabelID,"Bar Code","22",0,0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label
Success = GalleryApp.LabelClose(LabelID)
GalleryApp.Quit

Set GalleryApp = createObject ("LabelGalleryPlus.Application")
LabelID = GalleryApp.LabelOpen("sample3.lbl")
Success = GalleryApp.LabelSetVar(LabelID,"Bar Code","33",0,0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label
Success = GalleryApp.LabelClose(LabelID)
GalleryApp.Quit

**SLOW:**

*Description:*
This is a faster option than the slowest one but it is still slow. The main difference between slowest one and this is that object LabelGallery is created just once. Reason for the slowness opening and closing the label.

*Syntax:*

Set GalleryApp = createObject ("LabelGalleryPlus.Application")
LabelID = GalleryApp.LabelOpen("sample3.lbl")
    Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
    Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label
    Success = GalleryApp.LabelClose(LabelID)

LabelID = GalleryApp.LabelOpen("sample3.lbl")
    Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
    Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label
    Success = GalleryApp.LabelClose(LabelID)
LabelID = GalleryApp.LabelOpen("sample3.lbl")
Success = GalleryApp.LabelSetVar(LabelID,"Bar Code","33", 0,0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label
Success = GalleryApp.LabelClose(LabelID)

GalleryApp.Quit

FAST:

Description:
This is one of the faster options. Label opening and closing is performed only once.

Syntax:
LabelID = GalleryApp.LabelOpen("sample3.lbl")

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "33", 0, 0)
Success = GalleryApp.LabelPrint(LabelID, 1)    ' Print 1 label

Success = GalleryApp.LabelClose(LabelID)

FASTEST:

Description:
Label opening and closing is performed only once and LabelSessionStart command is used. The main reason that this is the fastest option is in LabelSessionStart command.

See also:
LabelSessionStart
**Syntax:**

LabelID = GalleryApp.LabelOpen("sample3.lbl")

GalleryApp.LabelSessionStart

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "11", 0, 0)
Success = GalleryApp.LabelSessionPrint(LabelID, 1)    ' Print 1 label

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "22", 0, 0)
Success = GalleryApp.LabelSessionPrint(LabelID, 1)    ' Print 1 label

Success = GalleryApp.LabelSetVar(LabelID, "Bar Code", "33", 0, 0)
Success = GalleryApp.LabelSessionPrint(LabelID, 1)    ' Print 1 label

GalleryApp.LabelSessionEnd

Success = GalleryApp.LabelClose(LabelID)
5. Glossary

5.1 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.

5.2 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.

5.3 Client/Server

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.
5.4 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.

5.5 Distributed COM

Distributed COM (DCOM) is a 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.

5.6 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.

5.7 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.

5.8 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.
5.9 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.