

NiceLabel 2017 User Guide for Designers

Rev-1601 ©NiceLabel 2016.

Contents

Contents	2
Introduction	4
Designer Terminology	4
System Requirements	5
Get Started	7
Workspace Overview	7
Configuring the Program	37
Keyboard and Mouse Support	41
Label	44
Label Setup Wizard	44
Label Properties	46
Label Objects	51
Selecting and Setting up a Printer	79
Working with Objects	80
Barcode	82
Source	82
Barcode	82
Check Digit	82
Human Readable	83
Bearer Bar	83
Details	84
Position	84
Relative Position	84
General	85
1D Barcode Details	86
2D Barcode Details	86
GS1 DataBar Specifics	88
Maxicode Barcode Content	88
Printing	90
Step 1: Create	90

Step 2: Preview	90
Step 3: Select printer	90
Step 4: Set print quantity	90
Step 5. Start Printing	91
Preview and Print a Label	91
Customize Printing Form	91
Printing Using NiceLabel Print	91
Store/Recall Printing Mode	92
Optimize Printing Speed	93
Handle Missing Images	94
Printing from Databases	94
Changing Common Printer Settings	95
Changing Dithering Options	96
Defining Unprintable Area	97
Work with Dynamic Data Sources	99
Variables as Dynamic Data Source	99
Functions as Dynamic Data Source	117
Databases as Dynamic Data Source	127
Internal Variables as Dynamic Data Source	160
Global Variables as Dynamic Data Source	161
Groups of Permitted Input Characters	164
Forms and Solutions	165
Solution	165
Form	166
Form Objects	168
Define Actions	205
NiceLabel Print	256
Managing Document Locations	256
Opening the Documents	256
Online Support	257

Introduction

Designer Terminology

This section describes the Designer elements that enable you to efficiently design a simple label or to create and manage a complex labeling solution that includes multiple labels, dynamic data sources and automatically run actions.

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Below listed are the essential Designer concepts. Being familiar with them gives a perfect starting point for successful labeling projects.

- [Solution](#)
- [Label](#)
- [Form](#)
- [Object](#)
- [Design Surface](#)
- [Content Providers](#)
- [Solution Explorer](#)
- [Data Source Explorer](#)
- [Actions Editor](#)
- [Dynamic Data Manager](#)

If you come across any other unfamiliar items while working with NiceLabel Designer, browse the [Help tab](#).

Label

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Label is the base of any designing and printing process in NiceLabel Designer. A label can be set as a simple file that marks an item with fixed content or a complex and thoroughly designed mixture of multiple dynamic product identifiers.

To design a printable label belongs to basic Designer tasks. Designer allows creating and printing of standalone labels and labels that are included in a printing [solution](#).

Read about how to create, design or edit a label [here](#).

Form

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

NiceLabel Designer form serves as a panel for entering, viewing and selecting the data to be presented and printed on a label. The advantage of using a form are simplified data-entry and label printing process for the end-user.

In NiceLabel Designer, a form is created within a printing solution. This means that a form is usually built in combination with a predesigned label.

Read about how to create, design or edit a form [here](#).

Solution

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

NiceLabel Designer solution is a single label printing file that includes multiple standalone items or interconnected labels and/or forms.

A solution enables adding any number of labels, forms and common [variable data sources](#). By doing this, a single Designer solution file serves as a container that envelops multiple labels and forms.

How do labels and forms cooperate in a solution? A label alone can be designed, printed, and, if necessary, reprinted. Multiplied manual printing of a single label file is time consuming and difficult if the content needs to be constantly updated. Therefore, NiceLabel introduced the ability to create forms which are combined with labels in a complete printing solution file.

In a solution, labels specify the layout of printed labels. Forms make sure the content of printed labels is easily defined, edited, updated, and reprinted. Forms also offer the user the control over a wide range of data- and print-related actions.

The advantages of keeping multiple labels and forms in a single file are:

- simplified management of printing solutions
- simpler and time efficient label designing and printing
- simplified use of variable data sources

Read about how to create or edit a solution [here](#).

Object

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

In NiceLabel Designer, an object is the basic building block for designing labels and forms. To design a label or form means to select, add, and position the objects on the [design surface](#).

EXAMPLE Each object performs a different role. For example, [Text](#) object is used for single-line textual content that does not need to adapt its font size to the label design. [Data initialization](#) object on the other hand serves a panel for assigning initial values to variables on the selected label.

Label object types and their purpose are listed [here](#).

Form object types and their purpose are listed [here](#).

System Requirements

NiceLabel 2017

- CPU: Intel or compatible x86 family processor
- Memory: 2 GB or more RAM
- Hard drive: 1 GB of available disk space

- 32-bit or 64-bit Windows operating systems: Windows Server 2008 R2, Windows 7, Windows 8, Windows 8.1, Windows Server 2012, Windows Server 2012 R2, Windows 10
- Microsoft .NET Framework Version 4.5
- Display: 1366×768 or higher resolution monitor

Additional Requirements For NiceLabel Control Center And NiceLabel PowerForms Web Components

- Database Server: Microsoft SQL Server 2005, Microsoft SQL Server 2008, Microsoft SQL Server 2012, Microsoft SQL Server 2014 (Express Edition of products listed above are also supported)
- IIS 7 or IIS 7.5 or IIS 8

Get Started

Read the following sections to make your first steps with Designer easy and efficient:

- [Workspace basics](#) section explains the Designer interface – menus, managers, dialogs and design surface.
- [Configuring the Program](#) section describes how the Designer can be set to adapt to the current needs while designing a label.
- [Keyboard and Mouse support](#) gives advice on efficient work with mouse and keyboard.

Workspace Overview

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Designer's workspace offers a flexible and easy-to-use environment for both – simple label designing and complex solution building.

Designer workspace follows the widely used application interface guidelines and is therefore equipped with tools and interface elements are familiar to a majority of users.

The Designer works space consists of the following segments:

- [Top segment: Tabs and Ribbons](#)
- [Left segment: Toolbars](#)
- [Right segment: Object Properties](#)
- [Design Surface](#)
- [Status and Printer bar](#)

Landing Page

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Designer's landing page is an introductory application page which opens once the software completes loading. It consists of the following segments:

- **Recent Files:** List of recently used Designer files.
- **Open other files:** Opens existing label and solution files.
- **New document area:** Enables you to create new Designer documents:
 - **New Label:** Creates a basic label for printing.
 - **New Solution:** Creates an advanced labeling solution with labels and forms.
 - **New from Sample Templates:** Creates a document based on a selection of industry standard compliant templates.

Tabs And Ribbons

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

NiceLabel Designer uses a standard Windows based interface.

The Designer's top section interface segments are described below.

Tabs

Tabs represent subsets of Designer features. The tabs contain interrelated commands that are available to the user in an organized way – grouped, and labeled:

- [File](#) (background): opens the print form and document management panel.
- [Home](#): offers commonly used commands such as copy/paste, print, and style commands.
- [Data](#): offers data source related commands.
- [View](#): gives you control over layout tools, zooming options and element markers visibility.
- [Solution](#): allows adding new labels and forms, starts printing actions and enables label file importing and exporting.
- [Contextual tabs](#): appear after clicking an object. They allow you to define object-specific settings. The type of contextual tabs adapts to the selected object.
- **Help**: besides offering the access to F1 help, this tab leads you to multiple helpful resources that make your work with Designer easier and more efficient.

Ribbon

Ribbon is a rectangular area that spreads across the top of an application window. Related commands are divided into ribbon groups. The ribbon changes along with the selected tabs and adapts to the currently used tools using the contextual tabs.

The following Designer dialog boxes are equipped with a dedicated ribbon:

- [Dynamic Data Manager](#)
- [Actions Editor](#)
- [Rich Text Editor](#)

File Tab

File tab serves as document management panel. The below listed options are available:

- [Info](#): defines the label or solution properties and prevents the label or solution from unwanted editing.
- [New](#): creates a new standalone label or a complete solution.
- [Open](#): allows opening existing label and solution files.
- [Import](#): allows you to import labeling files from non-NiceLabel labeling software.
- [Save](#): saves the active label or solution.

- [Save as](#): allows saving the active label or solution file by defining its name and location.
- [Print](#): opens the printing form.
- [Store](#): stores the current label as a template on the printer to be used in store/recall mode.
- Protection
- **Close**: closes the current Designer document.
- Options: opens the dialog for configuring the program defaults.
- **About**: provides license and software version information.
- **Exit**: closes the application.

New

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

New Label creates a new standalone label. [New Label Setup Wizard](#) opens after clicking this button.

New Solution creates a complete solution including (multiple) labels and printing forms. Solution designer opens after clicking this option.

New from Sample Templates creates a document based on a selection of industry standard templates.

NOTE Adding new labels or forms is also available in the **Solution explorer**. See section [Solution explorer](#) for more details.

Open

Open dialog allows opening existing label and solution files.

Browse allows selecting the label or solution files on local or connected network drives.

Recent Files field lists the latest files that have been edited. Click any of them to open the file.

File Tab Import

Import allows you to import files from NiceLabel labeling software.

Currently supported label formats are:

- Solution file (.nsln)
- Label file (.nbl)
- Label file (V6) (.lbl)
- XFF Form File (.xff)

When an import command is issued, the **Open** dialog opens. Select the label file for importing. If the import as finished successfully, a new unnamed label opens.

Save

Save saves the active label or solution using the same file name that was used for opening it.

NOTE If a file has been opened for the first time, **Save** directs you to the **Save as** backstage dialog.

Save as

Save as allows saving the active label or solution file by defining its name and location.

Recent folders field lists the folders that were recently used for saving the label or solution files.

Print

Print opens the print dialog. In Designer, the print dialog is represented by a customizable [printing form](#).

Details about the Designer print dialog are available [here](#).

Customize printing form options are described [here](#).

Store/Recall Printing Mode

Store/Recall printing mode optimizes the printing process. It increases printer response by reducing the amount of data that needs to be sent during repetitive printing tasks.

With store/recall mode activated, Designer does not need to resend the complete label data for each printout. Instead, default labels (templates) are stored in the printer memory and the Designer only sends recall commands which complete the stored label content during the printing process. Typically, a few bytes of data are sent to the printer, compared to a few kilobytes as would be the case during normal printing.

The action consists of two processes:

- **Store label.** During this process, Designer creates a description of the label template formatted in the selected printer's command language. When done, Designer sends the created command file to the printer memory and stores it.
- **Recall label.** A label stored in the printer memory is printed out immediately. Using the recall process, Designer creates another command file to instruct the printer which label from its memory should be printed. The recall label command occupies a few bytes of data only. The actual amount of data depends on the current situation. For fixed labels without any variable contents, the recall command file only contains the recall label command. For variable labels that contain variable fields, the command file includes the values for these variables and the recall label command.

NOTE Before activating this mode, make sure the appropriate printer driver is selected for the label printer. Not all label printers have the ability to use the store/recall printing mode.

Follow these steps to activate the **Store/Recall** printing mode:

1. Double click the label design surface. **Label Properties** dialog appears.
2. To enable the mode, select **Use store/recall printing mode** on **Printer** tab. Click **OK**.
3. Define the label template(s). All label objects with variable content must be formatted as internal printer objects:
 - Format the text object with internal printer fonts (not Truetype!).
 - Format barcode objects as internal printer barcodes.
 - If using variable objects formatted in Truetype fonts, variable pictures or database fields, default values are sent to the printer during the label store process.
4. Click **File -> Store**. Make sure the **Store variant** points to the correct memory location in the printer.
5. Insert or select the values for variable objects that are not formatted as internal printer objects. These variables will be given the same value on each label. They will behave as objects with fixed values.
6. Click **Store to printer** to create the command file with label template description and to send it to the printer.
7. Insert the values for prompted label variables. These variables link to the internal printer objects on the label. For this reason, their values can be changed during each printing.
8. Click **Print** to send the variable values and the recall label command to the selected label printer.

Protection

Protection prevents the label or solution from editing. This options is also useful for protecting the solutions from being copied or used in any unwanted scenarios.

- **Prevent document from editing:** locks the label or solution. Enabling this option makes the file uneditable – only printing is possible. A user may still disable this option and continue editing the file.
- **Requires a password to unlock document:** prevents the file editing with a password protection. Check this option to set a password. A file becomes editable only after entering a correct password.

Configuring the Program

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

To customize the general program configuration of Designer, open the **Options** dialog which is accessible from the **File** tab.

Designer configuration options are grouped on the following tabs:

- **Folders:** allows you to set the default locations for storing the labels, forms (solutions), databases and picture files.

- [Language](#): selects user interface language. Select the preferred language from the listed options. Designer interface language changes after the restart.
- [Global Variables](#): storage location for [global variables](#).
- [Printer usage](#): locally logged usage of installed printers.
- [Automation](#): enables you to configure NiceLabel Automation settings.

Home Tab

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Home Tab provides access to frequently used commands and settings in the following ribbon groups:

- [Clipboard](#): temporarily stores the selected elements, objects or groups of objects.
- [Font](#): group lets you define the font properties.
- [Action](#): group contains the **Print** button which starts the printing procedure or runs a form.
- **Management**: group provides direct access to the [Dynamic Data Manager](#) and Document properties – active [label](#) or [form](#) properties dialog.
- **Object**: group allows you to [align](#), group or [arrange](#) label objects.

Clipboard

Clipboard group temporarily stores the selected elements, objects or groups of objects. Use the selected and stored objects to transfer them from one label or solution to another.

TIP: Copying and pasting of textual (plaint text, RTF) and graphical (bitmaps) content between multiple applications is supported.

- **Paste**: pastes the clipboard contents on the design surface. Multiple reuse of a single clipboard item is allowed.
- **Cut**: removes the selected element(s) from the design surface and adds it to the clipboard to be pasted elsewhere. Note that the first element is selected by clicking it. When selecting additional elements, press and hold the <Shift> key while clicking these elements.
- **Copy**: copies the selected content to the clipboard. Multiple objects can be copied at once – select them and click **Copy**.
- **Delete**: deletes the selected elements or objects. They are not stored in the clipboard.

Font

Font group defines font properties:

- **Show/hide printer fonts:** button makes the printer fonts available on the typeface list or hidden.
- **Typeface:** defines the font family to be used in a selected object.
- **Point Size:** defines the text size in an object. Select the desired point size from the drop down selector or enter it manually.
- **Font Style:** defines the object text stylistic characteristics of text, such as bold or italic.
- **Alignment:** defines horizontal text positioning in an object: **Left**, **Center** or **Right**.
- **Justify:** makes a paragraph aligned along the left and right object margins.
- **Show/hide True Type and Open Type Fonts:** toggles TrueType and OpenType font visibility on the font list.
- **Show/hide Printer Fonts:** lets you toggle the visibility of fonts that are installed on the connected printers.

NOTE To set the default Text and Text Box object fonts, click the **Font** dialog box launcher at the bottom right part of the **Home** ribbon's **Font** group.

Action

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Action group starts the printing procedure or runs a form.

- **Print:** starts the Designer**Print dialog** as defined by the [Default Printing Form](#).
 - **Print:** opens the default printing form which serves as the Designer print dialog.
 - **Customize Printing Form:** allows adding, removing, rearranging or editing the printing form objects and their properties. Read more about the printing form customization [here](#).
 - **Recreate Printing Form:** resets the printing form to its default layout and reestablishes the dynamic content providers after being edited.
 - **Prompt Order:** allows changing the order in which the [variable](#) values are [prompted](#) at print time.
- **Run Form:** activates the print dialog based on the selected (active) label or form.

Management

Management ribbon group provides direct access to:

- [Dynamic Data Manager](#) dialog. Click the button to start managing the [dynamic data sources](#) that are connected to objects.
- **Document Properties** opens current [label](#) or [form](#) properties.

Object

Object group allows you to set:

- [Object alignment](#): positioning of object according to the design surface and other existing objects.
- [Object grouping and arranging](#).

Data Tab

Data tab displays the Designer ribbon with groups that enable you to instantly connect an object with commonly used data sources, or to define data connections in more detail:

- [Step-by-Step Database Wizard](#) ribbon group opens database wizard for typical database types.
- **Data Source Management** ribbon group gives direct access to the [Dynamic Database Manager](#) and [Prompt Order](#) dialogs.
- **RFID** ribbon group gives direct access to [RFID Tag](#) dialog.

Variables as Dynamic Data Source

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Variables serve as containers for storing and passing data between objects, scripts, external applications, printers, and user inputs. You may want to print labels on which data changes for each label. For example, counters, serial numbers, date and time, weight, article pictures... To accommodate the changing data, the labeling application can easily be used to format labels using variable data.

Designer offers multiple types of variables:

- [Variable](#): type of variable that changes its value at print time or according to user-defined conditions.
- [Variable Keyboard Input](#): type of variable that enables the content of a prompted field to be different for every print job. Its value is defined before each printing.
- [Current Date](#): current date taken as a variable value.
- [Current Time](#): current time taken as a variable value.
- [Counter](#): variable that changes its value incrementally or decrementally with each label print.

TIP: All label or solution variables are managed in [Data Source Explorer](#).

Step-by-Step Database Wizard

[Database wizard](#) is a guided process that allows the user to configure a connection to a database and to select which tables and fields will be used. Dedicated buttons provide instant access to the most commonly used database types. Use the **All Databases** button to start the wizard in general mode and to select the database type during the next step.

[Edit Database](#) allows you to edit all existing connected databases using a wizard.

The wizard additionally allows you to sort, filter records, and to define how many label copies will be printed per database record.

Data Source Management

Data Source Management ribbon group provides access to:

- [Dynamic Data Manager](#): dialog for managing and connecting to various data sources.
- [Prompt Order](#): dialog for defining the order of prompted variables on the print form.

View Tab

View Tab gives you control over document zooming, marker visibility, visual aids and design surface rotation. It makes the following ribbon groups available:

- [Zoom](#): defines design surface zoom level and Designer window zoom behavior.
- [Object Markers Visibility](#): defines visibility settings for object properties.
- [Alignment and Gridlines](#): sets object positioning behavior and defines properties for design surface gridlines.
- [Rotation](#): rotates the design surface clockwise for 90° per click.

Zoom

Zoom group defines the design surface zoom level and window zoom behavior. The exact **Zoom level** value and zoom behavior type are shown in the bottom right section of the Designer window.

- **Zoom to Document**: displays the entire label in the Designer window.
- **Zoom to Objects**: displays all objects in the Designer window.
- **Zoom in**: magnifies the design surface for a percentage of the currently defined zoom level.
- **Zoom out**: decreases the design surface for a percentage of the currently defined zoom level.

Object Markers Visibility

Objects markers visibility group toggles the visibility for the following object properties:

- **Object name**: displays the name of an object.
- **Internal element**: indicates if an object is connected to an [internal variable](#).
- **Counter**: indicates that the connected variable is a [Counter](#).
- **Locked object**: indicates that the object's position is locked.
- **Events**: indicates that the form object runs assigned [Action\(s\)](#).
- **Data Source**: indicates that the object is connected to a [dynamic data source](#).

Alignment and Gridlines

Alignment and Gridlines group sets object positioning behavior and defines properties for design surface gridlines.

- **Display grid line guides:** makes the design surface grid dots visible.
- **Grid Size X:** defines horizontal distance between the grid dots.
- **Grid Size Y:** defines vertical distance between the grid dots.
- **Grid Offset X:** defines the horizontal offset of the grid from the design surface center.
- **Grid Offset Y:** defines the vertical offset of the grid from the design surface center.
- **Align to Objects:** makes an object align with other object on the design surface. When an object is aligned, a line which marks the object alignment appears.
- **Align to Guides:** aligns the selected objects with gridlines.
- **Do Not Align:** makes the object position independent of gridlines and position of other object(s).

Rotation

Rotate view button rotates the design surface clockwise for 90° per click. Horizontal and vertical rulers adapt to the current position of the design surface.

Solution Tab

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Solution Tab enables quick and easy access to commands that are related to the entire print solution. The tab makes the following ribbon groups available:

- **Clipboard:** stores selected objects or groups of objects.
- **New:** allows adding additional labels or forms to the active solution.
- **Action:** starts the printing procedure or runs a form.
- **Import&Export:** allows importing, publishing and exporting the solution files.

Clipboard

Clipboard group temporarily stores the selected elements, objects or groups of objects. Use the selected and stored objects to transfer them from one label or solution to another.

TIP: Copying and pasting of textual (plaint text, RTF) and graphical (bitmaps) content between multiple applications is supported.

- **Paste:** pastes the clipboard contents on the design surface. Multiple reuse of a single clipboard item is allowed.

- **Cut:** removes the selected element(s) from the design surface and adds it to the clipboard to be pasted elsewhere. Note that the first element is selected by clicking it. When selecting additional elements, press and hold the <Shift> key while clicking these elements.
- **Copy:** copies the selected content to the clipboard. Multiple objects can be copied at once – select them and click **Copy**.
- **Delete:** deletes the selected elements or objects. They are not stored in the clipboard.

New

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

New allows adding additional labels or forms to the active solution. Labels and form that are included in the solution are listed in the [Solution explorer](#).

- **New Label:** adds a new label to the active solution. After clicking the **New Label** button the [Label Setup Wizard](#) appears.
- **New Form:** adds a new form to the active solution. After clicking the **New Form** button, a blank design surface appears. The form is ready for editing.

Action

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Action group starts the printing procedure or runs a form.

- **Print:** starts the Designer**Print dialog** as defined by the [Default Printing Form](#).
 - **Print:** opens the default printing form which serves as the Designer print dialog.
 - **Customize Printing Form:** allows adding, removing, rearranging or editing the printing form objects and their properties. Read more about the printing form customization [here](#).
 - **Recreate Printing Form:** resets the printing form to its default layout and reestablishes the dynamic content providers after being edited.
 - **Prompt Order:** allows changing the order in which the [variable](#) values are [prompted](#) at print time.
- **Run Form:** activates the print dialog based on the selected (active) label or form.

Set as Startup Form sets the current form as your default Designer form.

Import and Export

Import and Export group allows importing, publishing and exporting the solution files.

- **Import into Solution:** locates the label or solution files and imports them into the active solution. After clicking the **Import into Solution**, an open file dialog opens. Browse for the file to be imported and click **Open**.

- **Export Label:** saves the label to disk and makes it available for use in another solution. After clicking **Export Label** the Export label dialog appears. Select a location to save the label to.

Help Tab

Help tab provides direct access to various resources that help you design and use labels and forms quickly and efficiently.

Help ribbon group includes buttons with links to the following resources:

- **Help:** Designer online help
- **User Guides:** online collection of NiceLabel user guides. The collection includes user guides for the entire product portfolio.
- **Training Videos:** NiceLabel collection of training videos.
- **Knowledge base:** online library of articles that describe many technical solutions, tips and solved issues for labels and printing solutions.
- **Sample files:** access to the collection of sample label files. Use them to get familiar with Designer and to explore software capabilities.
- **Technical support:** connects you with NiceLabel technical support department.

Product ribbon group includes links to.

- Software **About** page
- NiceLabel web page.

RFID

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

RFID group provides access to the **RFID Tag dialog**. This dialog allows you to select the appropriate RFID tag type, to define its content, and to configure which type of data is going to be encoded on the tag.

NOTE RFID functionality is available with installed NiceLabel printer driver.

RFID Tag dialog allows you to configure how the tag content is encoded in a tag:

- [Select the RFID tag type.](#)
- [Configure various tag settings related to its structure and programming.](#)
- [Insert and configure data fields.](#)

[Print RFID data fields as internal text or barcode objects](#) option allows you to read and print the RFID data fields on a label using objects with internal printer elements.

Tag

Tag tab of the **RFID Tag dialog** allows you to select which tag type is going to carry the encoded data and how the data should be written to the tag.

Tag group includes the tag type selection.

- **Tag type** drop down list offers the selection of available RFID tag types. The selection of tag types is automatically defined by the printer driver.

NOTE Select the printer (and the corresponding driver) for the label with RFID tag in the [status bar](#).

Usage group defines the **RFID Tag** data sources and how the data is written to the tag.

- **Write data to tag while printing:** enables or disables data writing to the RFID tag.

Disabled writing might be useful during the label designing process or during specific workflow phases.

- **Print RFID data fields as internal text or barcode objects** option allows you to read and print the RFID data fields on a label using [Text](#) or [Barcode](#) objects with internal printer elements. Available fonts and barcode types are defined by the selected printer driver.

The encodable RFID data fields are added to the [Dynamic Data explorer](#) under **RFID Tag**.

DATA FIELD POSSIBILITIES

- **EPC:** data field with Electronic Product Code
- **User Data:** data field with payload to be encoded on the RFID tag.
- **TID:** data field with unique ID of the RFID tag.
- **GID Code:** general identifier code for RFID tags.
- **CID Code:** card identification number.

TIP: Drag the appropriate data field and place it on the label in form of a [Text](#) or [Barcode](#) object (defined by the driver).

Content

Content tab of the **RFID Tag dialog** allows you to define the content of an RFID tag. To encode the data in an RFID tag, complete the below described steps.

Step 1: Select Data Fields

Data fields group allows you to select the data fields. These fields are going to contain the encoded data of the RFID tag.

NOTE The selection of available Data Fields with corresponding settings depends on the selected [Tag type](#).

DATA FIELD EXAMPLES

- **TID:** unique ID of the RFID tag.
- **EPC:** syntax for unique identifiers assigned to objects, unit loads, locations, or other entities that are included in business operations.
- **User Data:** payload data to be written in the RFID tag.
- **RFID Tag Memory:** the only data field available for non-Gen2 RFID tags.

Step 2: Select Data Type

Data type defines the method for entering the **Data field** content. The availability of data types depends on the selected **Data field**:

- **Memory block:** the table allows you to enter the data into individual RFID tag memory blocks. Each table row represents a single block of the selected **Tag type**.

NOTE Memory block structure and properties of individual blocks depend on the selected **Tag type**.

NOTE **Data type** can be defined for each block individually.

- **Electronic Product Code (EPC):** added fields allow you to enter the RFID data according to the EPC standard.
- **ASCII string:** RFID data to be entered as a string of ASCII characters.
- **HEX encoded string:** RFID data to be entered as a string of hexadecimal pairs.
- **Numeric:** RFID data to be entered as a string of digits.

Step 3: Enter Value

Enter the value to be encoded in the RFID tag according to the selected **Data type**.

Security

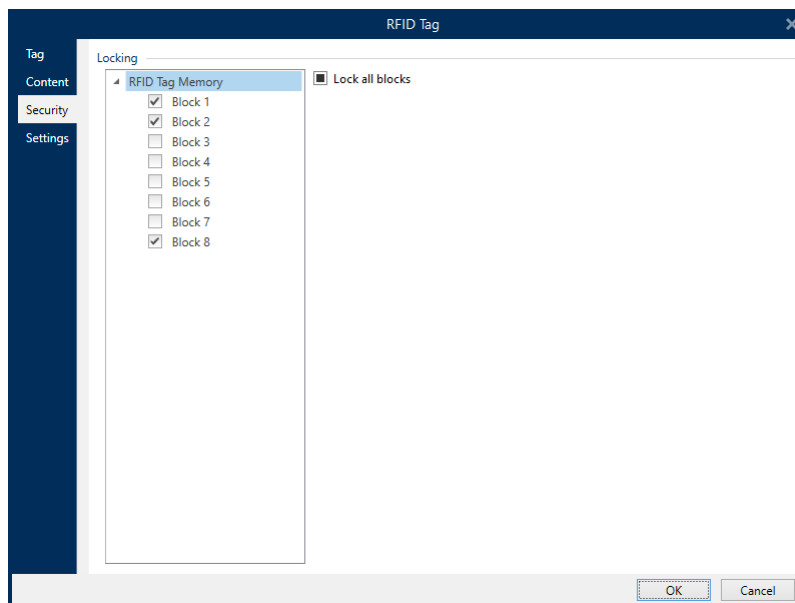
Settings tab of the **RFID Tag dialog** allows you to configure the RFID tag security settings. These settings allow or reject the access to RFID data writing or editing.

Security settings depend on the selected printer. There are three major configuration types.

Single Memory Field with Multiple Blocks

Locking group includes an overview of the blocks that are included in the RFID tag memory. Each block can be locked individually.

To protect the block for editing and writing, enable the **Block locked** option.



Lock all blocks option allows you to lock all blocks in the memory field simultaneously or unlock them if they are already locked.

Multiple Memory Fields

Access Protection group sets a password that must be entered before editing or writing the RFID data.

Data type defines the method for entering the **Password**.

- **ASCII string: Password** should be entered as a string of ASCII characters.
- **HEX encoded string: Password** should be entered as a string of hexadecimal pairs.
- **Numeric: Password** should be entered as a string of digits.

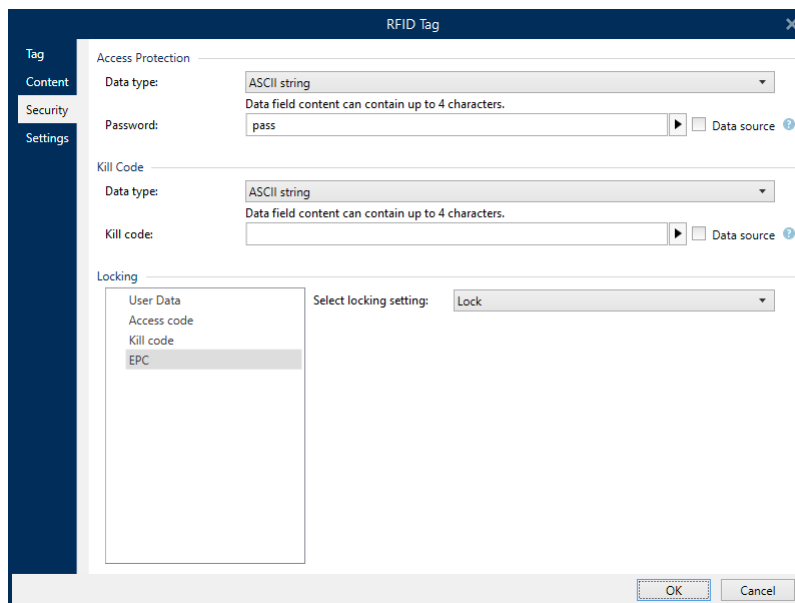
Kill code defines the code that disables the RFID tag permanently and irreversibly.

TIP: Having being activated, the data can neither be retrieved from nor written to the tag.

Data type defines the method for entering the **Kill code** characters.

- **ASCII string: Kill code** should be entered as a string of ASCII characters.
- **HEX encoded string: Kill code** should be entered as a string of hexadecimal pairs.
- **Numeric:Kill code** should be entered as a string of digits.

Kill code: code which permanently and irreversibly disables an RFID tag. Having being activated, the data can neither be retrieved from nor written to the tag.



Multiple Memory Fields with Block Locking

Additional settings from **Multiple Memory Fields** allow the user to set the locking for individual blocks withing RFID tag memory fields.

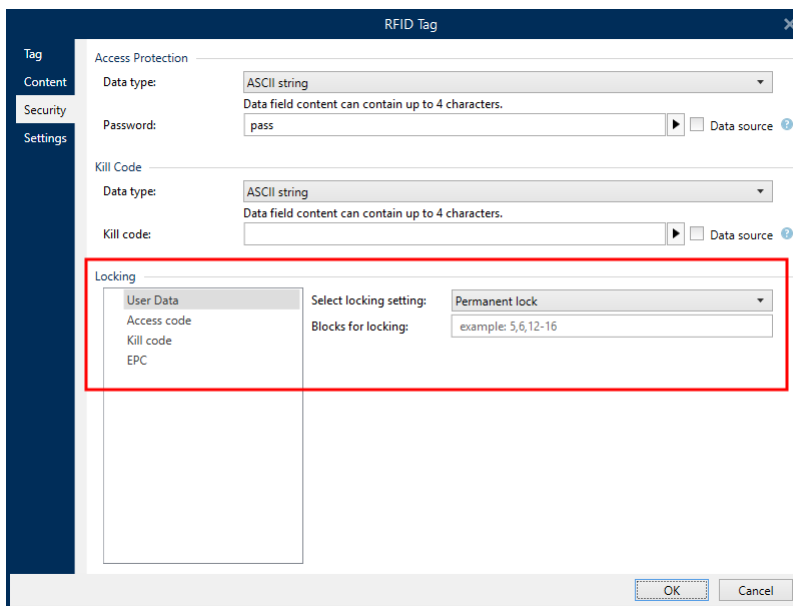
Locking group includes an overview of the memory fields that are included in the RFID tag. Each memory field can be locked individually.

Select locking settings allows you to define how the blocks are locked:

- **Preserve original locking setting:** original locking setting cannot be retrieved, but the default option assumes that the tag setting should remain unchanged.
- **Lock:** block is locked and further changes are prevented.
- **Unlock:** block is unlocked and editable.
- **Relock:** RFID tag is unlocked for the changes to be applied. When done, the tag is relocked immediately after.
- **Permanent lock, unlock or relock:** makes the above described settings permanent. These setting cannot be undone.

Blocks for locking: defines the individual blocks or range(s) of blocks to be locked.

TIP: Individually locked blocks are defined with an index and separated by a comma (with or without inserting the space between). Ranges of blocks are defined with a dash.



Settings

Settings tab of the **RFID Tag dialog** allows you to configure various tag settings related to its structure and programming.

All available settings are listed in a table under the **Settings** group:

- **Antenna offset:** defines distance between the top of the label and the embedded RFID inlay.
- **Power attenuation:** specifies radio output power. Use it to adjust RF emission power from the antenna.
- **Maximum tags to stop:** specifies how many tags are allowed to be programmed inadequately before the printing of labels stops. The option can be used as a precaution measure because it prevents endless consumption of labels. When the programming of the RFID tag fails, usually the word "VOID" is printed on the label.

- **Number of retries:** specifies the number of times the printer tries to program the tag if the initial attempt fails. The parameter is sent to the printer along with the rest of the data.
- **Check for valid tag:** before the tag programming begins, printer verifies if a proper RFID tag is available on the smart label. The printer also verifies if the tag is programmable.
- **Verify data write:** once the data has been encoded into the RFID tag, the printer checks if the written data is equal to the original value.
- **Electronic Article Surveillance (EAS):** is an anti-theft system used where an electronically-detectable tag is attached to the item.
 - **Preserve original EAS setting:** original EAS setting cannot be retrieved, but the default option assumes that the tag setting should remain unchanged.
 - **Enable EAS:** enables surveillance in the RFID tag. If this was the original setting, the tag would remain unchanged.
 - **Disable EAS:** disable surveillance in the RFID tag. If this was the original setting, the tag would remain unchanged.
 - **Permanently lock EAS tag setting:** permanently locks the chosen setting for the EAS. This lock cannot be undone.

NOTE The selection of available settings depends on the current **Tag type**.

RFID Read and Print

This section describes the procedure of defining which data fields from the RFID tag should be read and printed on the label using the internal printer elements.

Enable RFID Read and print

To enable the RFID read and print data functionality, open the [RFID Tag dialog \(Tag tab\)](#) and enable option **Print RFID data fields as internal text or barcode objects**. Currently available data fields are listed in the [Dynamic Data explorer](#).

Configure RFID Data Field Properties

To configure data field properties and to make it appear on the label, drag it to design surface. After adding it to design surface, the data field appears as a normal [Text](#) label object with below described additional properties.

Data format defines the format in which the RFID data field content is written in the label object and printed.

NOTE Available data formats and number of permitted characters are defined by the printer driver and selected tag type.

- **HEX encoded string:** data field content is a string of hexadecimal pairs.
- **ASCII string:** data field content is a string of ASCII characters.
- **Numeric:** data field content is a string of numbers.

NOTE The data field content is presented using internal printer elements. When selecting a font that does not belong to the internal printer fonts, the printing becomes impossible. An error appears.

Preview presents the data field content as it would appear using the selected **Data format**. Preview field does not include the actual encoded data. Enter the characters manually. By default, the object contains as many question marks, as given by the length of the RFID data field.

TIP: The role of **Preview** field is to fill the object with dummy content during the label design process and to give an impression of its layout on the printed label. The object on the actual printed label displays the content which was read from the RFID tag.

Data Extraction group defines which part(s) of data field content should be read from the RFID tag and printed on the label.

TIP: By default, the entire range of encoded data is read from the RFID tag.

- **Select bytes:** specifies which bytes of the encoded RFID tag data should appear in the label object.
 - **Starting byte:** the number of byte in an encoded string which starts the selection.
 - **Length in bytes:** number of selected bytes which should be extracted from the encoded data.
- **Select blocks:** specifies which blocks of the encoded RFID tag data should appear in the label object.
 - **Starting block:** the number of block in an encoded string which starts the selection.
 - **Number of blocks:** number of selected blocks which should be extracted from the encoded data.

Status And Printer Bar Intro

Status and printer bar offers instant printer selection and design surface zooming:

- [Printer Selection](#)
- [Design Surface Zooming](#)

Status Bar Printer Selection

Status Bar Printer Selection drop down list allows instant printer selection for label printing. The list is populated with printers which are installed on the system.

Design surface dimensions adapt to the selected printer automatically – as defined by the printer driver.

Zoom

Zoom group defines the design surface zoom level and window zoom behavior. The exact **Zoom level** value and zoom behavior type are shown in the bottom right section of the Designer window.

- **Zoom to Document:** displays the entire label in the Designer window.
- **Zoom to Objects:** displays all objects in the Designer window.
- **Zoom in:** magnifies the design surface for a percentage of the currently defined zoom level.

- **Zoom out:** decreases the design surface for a percentage of the currently defined zoom level.

Design Surface

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Design surface is Designer's central field that serves as a place to create, add, position, and interconnect the [label](#) and [form objects](#).

To make the designing of labels and forms as simple and efficient as possible, the design surface follows the same usability and functional principles as other standard Windows applications.

Design surface elements are described [here](#).

Design surface actions are described [here](#).

Design Surface Elements

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Below listed are basic elements that enable the user to interact with NiceLabel Designer.

Gridlines serve as a visual aid during the design process. They can be either visible or hidden. Their density is customizable. Gridline options are available in Designer's [Visual aids ribbon group](#).

Snaplines are non-visible alignment lines that help the user align the objects during the design process. Snap options are available in Designer's [Align ribbon group](#).

Ruler shows the available design area for label (white colored field) and file page (gray colored field).

Resize handles appear on the selected (active) objects. They enable you to resize the object dimensions. X and X dimensions can be resized simultaneously or separately.

Margins are the amount of fixed space between the edge of an object and the edge of a label.

Tabs for Active Documents allow the user to toggle between multiple labels and forms in a solution. Tabs are also used when designing [batches of labels](#) – header, body and tail labels are placed on separate tabs.

Design Surface Editing Actions

Below listed are the most relevant common actions for editing the objects on design surface:

- **Object layering:** allows the objects to be located in multiple layers. An object can be placed above or under the neighboring object. Layering options are described [here](#).
- **Objects aligning:** allows the objects to be aligned among each other. Aligning options are described [here](#).
- **Zooming:** enables the entire design surface to be zoomed in or out. Zooming options are described [here](#).
- **Scrolling:** enables sliding the design surface up and down.

- **Selecting:** enables the objects on design surface to be selected for editing individually or in a group. Group selection allows any actions to be applied to multiple object simultaneously.
- **Rotating:** enables object rotation.

Object Properties Toolbar

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

In addition to the object properties available in the Designer ribbon, the Properties toolbar opens on the right side of the design surface.

NOTE The Object properties toolbar is hidden at start and only appears after pressing F4 or clicking **Properties** on the right-click menu.

Available toolbar options adapt to each selected object and its properties:

- Available label objects and their properties are described [here](#).
- Available form objects and their properties are described [here](#).

Design Surface Elements

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Below listed are basic elements that enable the user to interact with NiceLabel Designer.

Gridlines serve as a visual aid during the design process. They can be either visible or hidden. Their density is customizable. Gridline options are available in Designer's [Visual aids ribbon group](#).

Snaplines are non-visible alignment lines that help the user align the objects during the design process. Snap options are available in Designer's [Align ribbon group](#).

Ruler shows the available design area for label (white colored field) and file page (gray colored field).

Resize handles appear on the selected (active) objects. They enable you to resize the object dimensions. X and X dimensions can be resized simultaneously or separately.

Margins are the amount of fixed space between the edge of an object and the edge of a label.

Tabs for Active Documents allow the user to toggle between multiple labels and forms in a solution. Tabs are also used when designing [batches of labels](#) – header, body and tail labels are placed on separate tabs.

Context Menus

In Designer, right mouse click displays various context menus that contain commonly used commands. The availability of commands depends on the selected items – design surface or object.

- Design surface context menu commands are described [here](#).
- Object context menu commands are described [here](#).

Design Surface Context Menu

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

When right-clicking the [design surface](#), a context menu appears. The context menu includes commonly used commands:

- **Document Properties:** opens the [label properties](#) or [form properties](#) dialog.
- **Paste:** pastes clipboard contents on the design surface. Multiple reuse of a single clipboard item is allowed.
- **Cut:** removes the selected element(s) from the design surface and adds it to the clipboard to be pasted elsewhere.
- **Copy:** copies the selected object to the clipboard.
- **Align with objects:** makes the object on the design surface align with other objects. When two objects are aligned, a leading line appears linking the edges of the two aligned objects.
- **Align with gridlines:** makes the object on the design surface align with gridlines. When moving the object, it always snaps to the gridline.
- **Display gridline:** makes the gridlines visible.
- **Select all:** selects all object on the design surface.
- **Objects markers visibility:** toggles visibility for the below listed object properties. Markers become visible when moving the mouse pointer over the object:
 - **Object name:** marker shows the name of an object.
 - **Internal element:** marker shows if the selected object belongs to the internal printer elements.
 - **Counter:** marker shows that the connected variable is [Counter](#).
 - **Locked object:** marker shows that an object's position is locked.
 - **Events:** marker shows that the form object runs assigned [Action\(s\)](#).
- **Zoom:** defines zooming behavior:
 - **Zoom to Document:** shows the entire label in the Designer window.
 - **Zoom to Objects:** shows all objects in the Designer window.

Object Context Menu

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

When right-clicking an object, a context menu appears. The context menu includes the below described commands:

- **Properties:** opens the [label properties](#) or [form properties](#) dialog.
- **Copy:** copies the selected content to the clipboard

- **Cut:** removes the selected element(s) from the design surface and adds it to the clipboard to be pasted elsewhere. Note that the first element is selected by clicking it.
- **Delete:** removes the selected object from the design surface.
- **Lock position:** prevents the selected object from being moved.
- **Arrange:** positions the objects so that they appear either in front of or behind each other:
 - **Bring Forward:** sends the element forward for one level.
 - **Send backward:** sends the element back for one level.
 - **Send to Front:** sends the element in front of all other elements on the label.
 - **Send to Back:** sends the element behind all other elements on the label.

Group Context Menu

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

When right-clicking an object, a context menu appears. The context menu includes the below described commands:

- **Document Properties:** opens the [label properties](#) or [form properties](#) dialog.
- **Copy:** copies the selected content to the clipboard
- **Cut:** removes the selected element(s) from the design surface and adds it to the clipboard to be pasted elsewhere. Note that the first element is selected by clicking it.
- **Delete:** removes the selected object from the design surface.
- **Select All:** selects all added objects on a label or form.
- **Alignment and Gridlines**
 - **Align to Objects:** makes an object align with other object on the design surface. When an object is aligned, a line which marks the object alignment appears.
 - **Align to Guides:** aligns the selected objects with gridlines.
 - **Do Not Align:** makes the object position independent of gridlines and position of other object(s).
 - **Display grid line guides:** makes the design surface grid dots visible.

Objects markers visibility group toggles the visibility for the following object properties:

- **Object name:** displays the name of an object.
- **Internal element:** indicates if an object is connected to an [internal variable](#).
- **Counter:** indicates that the connected variable is a [Counter](#).

- **Locked object:** indicates that the object's position is locked.
- **Events:** indicates that the form object runs assigned [Action\(s\)](#).
- **Data Source:** indicates that the object is connected to a [dynamic data source](#).
- **Zoom:** defines zooming behavior:
 - **Zoom to Document:** shows the entire label in the Designer window.
 - **Zoom to Objects:** shows all objects in the Designer window.
- **Group Objects:** unites the selected objects and make them behave as a single element.

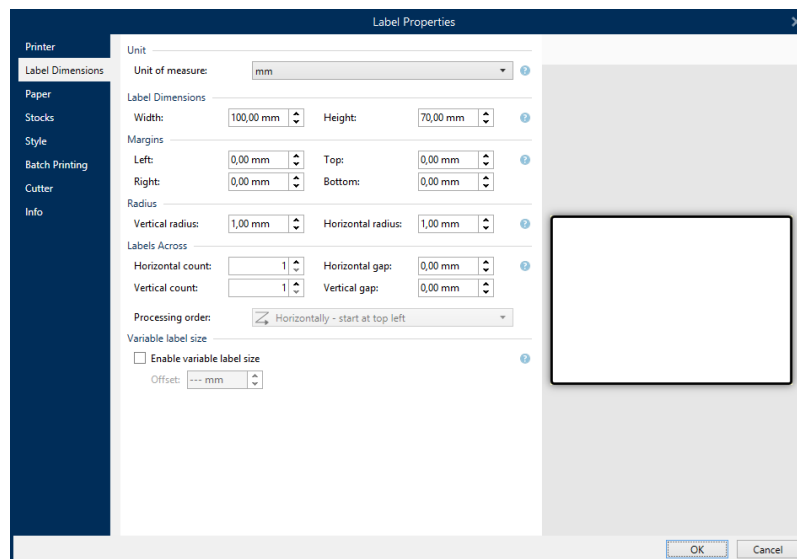
Document Properties And Management Dialogs

Label Properties

Label Properties dialog selects the printer, sets label dimensions and defines the printing paper properties.

The settings are available on the below listed dialog tabs.

Label Property	Description
Printer	Selects the preferred printer.
Label Dimensions	Defines the Unit of measure and label dimensions.
Paper	Defines the printing paper properties.
Stocks	Selects the stock type.
Style	Defines the label style parameters.
Batch Printing	Defines details for grouped printing of labels.
Cutter	Enables label roll cutting during or after the printing procedure.
Info	Inserts the label description.



TIP: To open the **Label Properties** dialog, double click the [design surface](#).

Printer

Printer tab lets you define the printer to print the labels on, and sets basic printing behavior.

Printer drop down menu selects a printer from the currently installed printers.

TIP: To set the printer settings, select a printer and click **Printer properties**. This button gives direct access to the selected printer's driver and its settings.

NOTE For additional information on the installed printer drivers and their settings, read the [NiceLabel Driver Installation Manual](#).

- **Always use the default printer:** selects the default system printer to be used for the current print job.
- **Double-sided printing:** enables double-sided label printing.
- **Use custom printer settings saved in the label:** each label may have its own printer settings defined and saved by the user. Select this option to use these settings while printing.
- **Use default printer settings from the printer driver:** select if you prefer the default printer settings or if no custom settings have been defined. Default printer settings are going to be used for printing.

Printing group of settings optimizes the printing process.

- **Optimize printing of identical labels:** if multiple identical labels are printed, the printer does not need to receive the label file each time. With this option enabled, the printer alone multiplies the print job.
- **Use advanced printer driver interface:** speeds up label printing.

TIP: When selected, the optimized printer commands are in use. Deselected option disables printing optimization. Each label is sent to the printer in form of an image.

- **Combine all non-printer elements into a single graphic item when sent to a printer:** merges multiple items and sends it to the printer as a single printable graphic.
- **Use store/recall printing mode:** optimizes printing performance.

With this mode activated, the Designer does not need to resend the complete label data for each printout. Instead, default labels (templates) are stored in the printer memory and the Designer only sends recall commands to complete the label content during the printing process. For more information, read section [Use Store/Recall Mode](#).

- **Store variant:** printer memory location to store the label templates.

NOTE To make sure the stored label samples are not lost after power cycling the printer, store them at non-volatile locations.

Label Dimensions

Label Dimensions tab specifies label dimensions and defines whether its size should adapt to the changing size of the objects or not.

Unit of measure defines the unit to be used while designing the label. There are four available units: cm, in, mm, and dot.

Label Dimensions group defines the label's **Width** and **Height**.

NOTE When manually inserting the unit of measure, this also changes the currently defined **Unit**.

Margins group sets the distance between the edge of the printing surface and the edge of the label (left/right, top/bottom).

Most laser and other non-thermal printers cannot print over the entire label surface. There is usually a non-printable label area of about 5 mm from the border of a page. In Designer, this area is marked by a red line. Any object on or beyond the red line is not printed entirely.

Radius group enables you to make the label corners rounded.

- **Vertical radius:** adjusts roundness value in vertical direction.
- **Horizontal radius:** adjusts roundness value in horizontal direction.

Labels Across defines the number of labels to be printed on a single label sheet.

- **Horizontal count:** labels distributed horizontally.
- **Vertical count:** labels distributed vertically.
- **Horizontal gap:** horizontal distance between labels on a sheet.
- **Vertical gap:** vertical distance between labels on a sheet.

Variable Label Size enables the label size to change in accordance with the size of its objects.

When assigning additional data to label objects, their size increases and occupies more space. Therefore, the label height must adapt.

- **Offset:** distance between the last object on a label and the bottom edge of a label.

Paper

Paper tab sets printing paper properties.

Unit selects the **Unit of measure** to be used in a label.

Paper Type group defines paper dimensioning type – automatic or manual.

- **Automatically set page size based on the label dimensions (labels on a roll):** page size is defined by the printer driver.

NOTE If a thermal printer is selected in the previous wizard step, this option is enabled by default.

- **Manually set page size (sheets of paper):** page size is set manually.

NOTE If a regular office laser printer is selected in the previous wizard step, this option is enabled by default.

In case the page size is defined manually, additional options appear:

- **Paper:** selection of standard paper formats.
- **Width** and **Height:** custom paper dimensions.

Orientation group sets the new label layout as **Portrait** or **Landscape**.

- **Rotated: Printer Layout** rotation for 180 degrees.

Preview displays current label screen and print layouts.

Stocks

Label stocks are a time-saving alternative to designing labels from scratch. Use stock templates when designing labels for a specific printer type and when optimizing the label designing process.

Stock type defines which stock type should be used when designing and printing a label. Stock types are usually associated with printer vendors or stationery suppliers.

NOTE Here defined stock properties override manually set label properties.

Stock defines the exact stock which belongs to the **Stock type**.

NOTE If the selected stock is not compatible with the selected printer, a warning appears. Label designing and printing becomes impossible.

Stock information displays the selected stock's properties:

- **Label dimensions**
- **Labels across**
- **Description**
- **Author**

Style

Style tab is used for defining label style parameters.

Background color is defined by the **Standard** or **Advanced** color selection.

Background picture sets a picture on the label background.

- **Embed the picture to document:** makes the picture an integral part of label document.
- **Save embedded picture to file:** embedded picture is saved to a file.
- **Picture position:** background picture to be centered, to fit the label dimensions, or to be stretched.
- **Rotation:** background picture rotation by 90 degrees.
- **Print background picture:** background picture is printed.

Batch Printing

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

Batch printing allows grouped printing of labels that belong to the same batch.

TIP: A batch is a set of labels printed within a single print job. Each print job can consist of a single or multiple batches.

The first purpose of batch printing is to automate the execution of a predefined action after the batch has been printed.

EXAMPLE Label roll is automatically cut after a batch of five labels has been printed.

The second purpose of batch printing is to enable header and tail label printing with each batch.

EXAMPLE A batch of five labels starts with a header and ends with a tail label. Both of them differ from the main (body) labels.

- **Enable batch printing:** activates batch printing mode. Batch definition menu becomes active.
- **Batch definition:** specifies what a batch of labels should consist of:
 - **All labels in the print job:** all labels in the current print job are assigned to the same batch.
 - **Batch ends after a specific number of labels:** batch is finalized after a specified number of labels is printed.
 - **Batch ends when the data source changes value:** changed value of the selected variable is used as a marker for opening a new batch.

Actions group defines the action that executes after a batch has been printed. The availability of actions depends on the selected printer's driver. If the driver provides no information on the action availability, the list is empty.

EXAMPLE Commonly used batch actions are **Cutter**, **Pause printer**, **Batch mark**, **Batch separator**, etc. With a defined web of labels (label template with labels next to each other), an applicable action also becomes **Eject page**. These printer commands can be applied dynamically during the printing process.

Header / Tail Labels group specifies the properties of header and tail labels in a batch.

- **Use header label:** header label of a batch.
- **Action after header label:** action to be taken after the header label has been printed. The selection of available actions depends on the selected printer's driver.

NOTE The selection of available actions depends on the selected printer's driver.

- **Use tail label:** last label of a batch.
- **Action after tail label:** action to be taken after the tail label has been printed.

NOTE The selection of available actions depends on the selected printer's driver.

TIP: Header, tail and main (body) labels of a single batch are accessible via tabs that are located under the design surface (gray field).

Cutter

Cutter enables label roll cutting during or after the printing procedure.

Enable cutter activates the label cutter and its settings.

Cutter mode specifies when the label roll is cut.

- **Cut after the last printed label:** label roll is cut after the finished print job.
- **Cut after a specific number of labels:** label roll is cut at a defined number of labels or if a defined condition is met.
- **Cut when the data source changes:** label roll is cut when the value of a data source changes or if a defined condition is met.

Info

Info tab includes a **Description** that serves as a hint or as a guidance for the user that is going to work with the label.

Define label **Description** by entering text into the field.

Form Properties

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

Form Properties dialog is used for defining various form properties.

TIP: To open the **Form Properties** dialog, double click the [design surface](#).

The settings are available on the below listed dialog tabs.

Form Property	Description
Basic Settings	Selects the preferred printer.
Additional Settings	Selects the stock type.
Style	Defines the Unit of measure and label dimensions.
Tab Order	Defines the order of focus shifting.
F1 Help	Defines the printing paper properties.
Events	Defines the label style parameters.
Variable Events	Enables label roll cutting during or after the printing procedure.
Info	Inserts the label description.

Basic Settings

Basic Settings tab is used to define the title, size and startup behavior of a form.

Title defines the form ID.

- **Show form title bar:** window title bar visible or hidden upon form startup.
- **Allow closing form:** form close using the window **Close** button allowed or not.

With this option disabled, the form can be closed from taskbar.

- **Allow resizing form:** form size customizable or not.

Disable this option to lock the form size.

Size group defines the form's **Width** and **Height**.

Initial form state group defines the form state upon startup.

- **Maximized:** form opens in full screen mode.
- **Default form size:** the form appears using the manually defined sizes.

Startup form position group defines the on-screen position of a form upon its startup.

- **As defined:** the form appears at a location defined by the distance in pixels from **Left** (left edge of the form) and **Top** (top edge of the form).
- **Screen center:** screen center is the startup form position.

Additional Settings

Additional Settings tab allows selecting the form scripting language. There are two scripting languages available for Designer form objects: **VBScript** and **Python**.

- **VBScript:** scripting for advanced data operations, comparisons and direct calculations on a form.
- **Python:** suitable for 64-bit systems. A significantly faster scripting alternative to **VBScript**.

Style

Set the label style parameters.

- **Background color** is defined by the **Standard** or **Advanced** color selection. Switch between these two options by clicking the **Advanced** or **Basic** button.
- Browse for the label **Background picture** or insert the direct path. Once the picture has been defined, it is possible to:
 - **Embed the picture to document:** makes the picture an integral part of label document.
 - **Save embedded picture to file:** embedded picture is saved to a file.
 - **Picture position:** background picture to be centered, to fit the label dimensions, or to be stretched.

Tab Order

Tab order tab customizes the order of setting the focus on form objects while pressing the **Tab** key.

- **ENTER key behaves as TAB key:** **Enter** key has the same role as the **Tab** key does.
- Select the form **Object** and move it up or down to define the focus switching order.

F1 Help

F1 Help tab defines custom form help content to help the end-user while designing and/or using a form.

TIP: Enter custom text in the editing field and click **OK**.

Events

Events tab allows setting actions for basic form events.

- **On Form Load:** the action is run upon form load.
- **On Form Close:** the action is run when the form is closed.
- **On Form Timer:** the action is run after a given time interval.
 - **Interval:** duration of the time interval.

TIP: Click [Actions ...](#) to set the actions that are run by the listed events.

Variable Events

Variable Events selects the variables that are monitored for changes in their values.

- **Add:** adds a [variable](#) to the list.
- **Delete:** removes a [variable](#) from the list.

TIP: Click [Actions ...](#) to set the actions that are triggered by changed values in the listed variables.

Info

Info tab includes a **Description** that serves as a hint or as a guidance for the user that is going to work with the form.

Define form **Description** by entering text into the field.

Dynamic Data Manager

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Dynamic Data Manager is a dialog that enables the user to [manage the dynamic data sources](#) for label and form objects.

[Label](#) and [form](#) objects can be connected to multiple variables, functions and databases.

To open the dialog, click the **Dynamic Data Manager** button in the Designer ribbon.

Dynamic Data Manager consists of the following segments:

- [Ribbon](#): includes editing commands, enables direct connection to databases, and offers access to the [Step-by-Step Database Wizard](#).
- [Data Source Explorer](#): enables adding new data sources and browsing the existing ones.

Explorer gives an instant overview of the data that is currently in use as dynamic content of label and form objects.

- **Main/editing field:** is used to define the data source properties. Its elements and layout adapt to current use.

Read more about how to define the data sources in the following sections:

- [Work with variables.](#)
- [Work with functions.](#)

- [Use databases as content source.](#)
- [Use internal variables as content source.](#)
- [Use global variables as content source.](#)

Configuring The Program

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

To customize the general program configuration of Designer, open the **Options** dialog which is accessible from the **File** tab.

Designer configuration options are grouped on the following tabs:

- [Folders](#): allows you to set the default locations for storing the labels, forms (solutions), databases and picture files.
- [Language](#): selects user interface language. Select the preferred language from the listed options. Designer interface language changes after the restart.
- [Global Variables](#): storage location for [global variables](#).
- [Printer usage](#): locally logged usage of installed printers.
- [Automation](#): enables you to configure NiceLabel Automation settings.

Folders

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Folders tab defines the default location for opening and storing the documents and files that are edited and used in Designer.

NOTE Make sure read/write rights are granted to the account under which the Designer is running on the computer.

- **Labels**: location for opening and saving the label files.
- **Forms**: location for opening and saving the forms.
- **Database**: location for file databases (Excel, Access, Text).
- **Picture**: location for opening the picture files.

Folders set in this tab serve as the default location when searching for a specific file in Designer.

EXAMPLE

File name: picture.png

Result: When searching for a graphic file named picture.png, Designer goes to C:\Users\user\Documents\NiceLabel\Graphics

Language

Language tab allows selecting the Designer interface language. Select the appropriate language and click **OK**.

NOTE Restart is necessary to make the user interface appear in the selected language. Make sure you save your work before closing the program.

Global Variables

Global Variables tab allows defining which location with stored [global variables](#) should be used:

- **Use global variables stored on the server (Control Center):** sets the global variable storage location on the Control Center.

NOTE Select the Control Center before selecting this option.

NOTE This option becomes available when using the NiceLabel Label Management Solution license.

- **Use global variables stored in a file (local or shared):** sets the global variable storage location in a local or shared folder. Enter the exact path or click **Open** to locate the file.

TIP: These two options become useful when designing solutions for multiple customers with their own sets of global variables.

Printer Usage

Printer usage tab displays the locally logged usage of installed printers.

TIP: Printer usage provides information about the print jobs which have been sent by this computer.

NOTE Printer usage logging is available with multi-seat license.

Printer usage information group displays how many of the permitted printer ports are used by printing on multiple printers.

- **Number of printers allowed by license:** number of permitted printers to be used with the current Designer license.
- **Number of used printers in the last 7 days:** number of printers that have been used with Designer during the last 7 days.

During a 7-day period, Designer license allows only the specified number of different printers to be used.

WARNING When exceeding the allowed number of printers – this number is defined by the license – a warning appears. When doubling the number of allowed printers, printing is no longer allowed.

Printing statuses are visible in multiple columns:

- **Printer:** name or model of the printer that was selected for the print job.

NOTE If the connected printer is shared, only printer model is displayed.

- **Location:** name of the computer from which the print job has been sent.
- **Port:** port used by the printer.
- **Last Used:** time passed since the last print job.
- **Reserved:** prevents the printer from being removed after idling for more than 7 days.
- **Remove:** removes the printer from the list of used printers.

Information about the removed printer is broadcasted. Use it when the number of used printers exceeds the number permitted by the license.

NOTE If a printer remains unused for more than 7 days, it is removed automatically unless the **Reserved** option is enabled.

Control Center

Control Center tab allows you to enable and configure the monitoring of events and print jobs. The use of Control Center enables centralized event and print job reporting, and centralized storage of global variables.

NOTE This tab is available only when LMS license is activated.

Address

Address group defines which Control Center server should be used.

- **Control Center server address:** URL of the connected Control Center server. You can select from the list of automatically discovered servers on the network, or enter a server address manually.

NOTE The license keys on the Control Center server and on the workstation must match to enable the connection.

Event Monitoring

Event handling in the Control Center allows central management of labeling workstation activities. Activities like label printing, errors, alerts, middleware application triggering, etc. are reported and logged to Control Center.

Event Monitoring group defines what types of events should be logged by the connected Control Center:

- **Print Events:** logs the print related events from the workstation.
- **Error Events:** logs all reported errors.

NOTE By default, Print Events and Error Events are logged to the Control Center.

- **Trigger Activity:** logs all fired triggers.
- **Trigger Status Change Events:** logs the trigger status changes which have been caused by the fired triggers.

Print Job Monitoring

Print Job Monitoring group enables you to log the completed and ongoing print jobs to the Control Center.

- **Enable Print Job Logging to Server:** activates print job logging.
- **Detailed printing control:** enables monitoring of statuses that are reported by the connected printer.

NOTE There are two requirements to make this option available:

- The printer must support bidirectional communication.
- NiceLabel printer driver must be used for printing.

Command Line Arguments

Automation

Automation tab enables you to configure NiceLabel Automation settings.

Service Communication

Service Communication group defines the communication settings.

- **Service communication port:** number of the port which is used by the Automation service for communication.

Log

Log group configures how the below listed messages reported by the Automation Manager are logged.

NOTE The default data retention time is 7 days. To minimize log database size on busy systems, reduce the retention period.

- **Clear log entries daily at:** selects the time at which the daily log entries are cleared.
- **Clear log entries when older than (days):** sets the log retention time in days.
- **Log messages:** selects the message types that are logged.
 - **All messages:** saves all message types in the log.
 - **Errors and warnings:** saves errors and warnings in the log.
 - **Errors:** saves errors in the log.
 - **No log:** no messages are logged.

Performance

Performance group enables improving the time-to-first label and general performance of the Automation service.

- **Cache remote files** enables storing the remotely located files in the local cache.
 - **Refresh cache files (minutes):** time interval within which the files in the cache are synchronized with the files in the original folder. This is the time interval which limits the system to use version which may not be the latest.
 - **Remove cache files when older than (days):** defines the time interval after which all files in cache are removed.

TIP: When loading the labels, images and database data from network shares, you might experience delays when printing the labels since Automation service has to fetch all required files before the printing process can begin. File caching solves this issue by storing the necessary files in the local cache.

Keyboard And Mouse Support

To efficiently perform and complete the Designer tasks, follow the guidelines related to the use of keyboard and mouse:

- [How to efficiently use keyboard and mouse](#)
- Keyboard support
- [Keyboard shortcuts](#)
- [Mouse wheel support](#)

How To Efficiently Use The Keyboard And Mouse

Use the below listed tip to make your work with Designer easier and more efficient.

1. **Select object anchoring point.** Press <CTRL> key and click the object placeholders to quickly define the anchoring point.
2. **Label scrolling and zooming.** Use mouse wheel to scroll the label. Holding <CTRL> when rotating the wheel, adjusts zoom factor. <SHIFT> scrolls label left or right.
3. **Set label or form properties.** Double click the design surface to open the [label](#) or [form properties](#) dialog.
4. **Vertical or horizontal object moving.** Hold <SHIFT> while moving an object over the design surface. The object is moved in straight vertical and horizontal lines.
5. **Resize an object with arrow keys.** Holding <SHIFT> while pressing arrow keys resizes the object.
6. **Fine tune the object position.** Hold <CTRL> while pressing arrow keys.
7. **Open contextual menus.** Right click the object or design surface to access the [label, form](#) or [design surface](#) contextual menus.
8. **Select multiple objects.** Hold <SHIFT> and click the objects to add them to the selected objects in a group.

9. **Quickly add an object with connected data source.** Click the object's shortcut handle in the [object toolbar](#). A list of available data sources appears. Select a data source or add a new one, and click the design surface to add an object which already has a dynamic data source connected to it.

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Mouse Wheel Support

Use mouse wheel to speed-up design object zooming and design surface scrolling.

- Turning the wheel scrolls the label vertical direction.
- Holding <SHIFT> and turning the wheel scrolls the label left or right.
- Holding <CTRL> and turning the wheel, zooms the label in or out.

Keyboard Shortcuts

Use keyboard shortcuts to reduce the time needed to accomplish frequent tasks with Designer. To complete these tasks, use a standard combination of keys.

TIP: Keyboard shortcuts are just a faster and more convenient way of choosing commands. The command itself is executed in the same way as if it was run from the menu or toolbar.

Action	Press
Open	CTRL+O
Save	CTRL+S
Close	ALT+F4
Cut	CTRL+X
Copy	CTRL+C
Paste	CTRL+V
Select all	CTRL+A
Bold	CTRL+B
Italic	CTRL+I
Center text	CTRL+E
Left align text	CTRL+L
Right align text	CTRL+R
Cancel	ESC
Undo	CTRL+Z
Redo	CTRL+Y
Zoom	CTRL+mouse scroll

Move Focus	TAB or SHIFT+TAB
Print	CTRL+P
Move left	←
Move right	→
Move up	↑
Move down	↓

Label

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Label is the base of any designing and printing process in NiceLabel Designer. A label can be set as a simple file that marks an item with fixed content or a complex and thoroughly designed mixture of multiple dynamic product identifiers.

To design a printable label belongs to basic Designer tasks. Designer allows creating and printing of standalone labels and labels that are included in a printing [solution](#).

Read about how to create, design or edit a label [here](#).

Label Setup Wizard

Label Setup Wizard guides you through the process of creating a new label. The wizard consists of four configuration steps and a summary:

- [Step 1: Select a Printer](#)
- [Step 2: Page Size](#)
- [Step 3: Label Layout](#)
- [Step 4: Label Dimensions](#)
- [Step 5: Summary](#)

After finishing these steps, the label is ready for editing and printing.

NOTE To quit the Label Setup Wizard during any step, press escape. The new label properties are set to default.

Label Setup Wizard

Step 1: Select The Printer

This step selects the printer to be used for printing the newly created label. It also provides direct access to printer driver properties.

Select the printer from the drop-down list. To set the printer settings, select a printer from the list of installed printers and click **Printer properties**. This button gives you direct access to the selected printer driver and its settings.

NOTE When changing the printer, [Page Size](#) settings always go to default (automatic).

- **Always use the default printer:** sets the default system printer to be used for the current print job.
- **Double-sided printing:** enables double-sided printing for the new label.
- **Preview field:** displays the label layout according to the currently set properties.

NOTE For additional information on the installed printer drivers and their settings, read the [NiceLabel Driver Installation Manual](#).

Step 2: Set The Page Size

This step defines how the page size is selected. When using a thermal printer, it is recommended to set the size automatically. Manual selection proves to be useful if you know the exact stock code or label format.

Print on a roll of labels option prints on the installed roll of labels. Page size for thermal printers is detected automatically.

NOTE If a thermal printer is selected in the preceding [Select the Printer](#) wizard step, this option is enabled by default.

Print on a sheet of paper option prints labels on sheets of paper. It lets you manually define the label page size to fit the printer.

With this option selected, additional settings appear:

- **Unit of measure:** defines the unit of measure to be used while designing the label.
- **Paper:** defines the label page **Width** and **Height**.

NOTE If a regular home/office printer is selected in the preceding [Select Printer](#) wizard step, this option is enabled by default.

Load settings from a predefined stock option sets the page to be defined by the selected stock type.

With this option selected, additional settings appear:

- **Stock type:** defines which stock type should be used when designing and printing the newly created label. Stock types are usually associated with printer vendors or stationery suppliers.
- **Stock:** defines the exact stock from the **Stock type**. Select it from the drop down list. All listed stocks belong to the same **Stock type**.

NOTE If the selected stock is not compatible with printer, a warning appears. Label designing and printing becomes impossible.

- **Stock information:** displays the selected stock's properties.

Step 3: Select Label Layout

This step defines the label orientation and rotation on a printer:

- **Orientation:** sets the new label layout as **Portrait** or **Landscape**.
- **Rotation:** rotates the **Printer Layout** of a label for 180 degrees.
- **Preview field:** displays the label layout according to the currently set properties.

Step 4: Specify Label Dimensions

This step defines the dimensions of the newly created label, its margins, measurement unit, and labels across positioning settings:

- **Unit of measure:** defines the unit to be used while designing the label.
- **Label Dimensions:** define the new label's **Width** and **Height**.

- **Margins:** define the distance between the edge of the printing surface and the edge of the label (left/right, top/bottom).
- **Labels Across:** defines the number of labels to be printed on a single label sheet.
 - **Horizontal count:** distributes labels horizontally.
 - **Vertical count:** distributes labels vertically.
 - **Horizontal gap:** sets horizontal distance between the labels on a sheet.
 - **Vertical gap:** sets vertical distance between the labels on a sheet.
- **Processing order:** defines the direction in which the labels are printed. Set the starting corner where the printing starts and define the horizontal and vertical direction of label positioning.

Step 5: Summary

This step summarizes the new label properties as defined using the **Label Setup Wizard**.

Before clicking **Finish** to enter the label editing and printing phases, check the displayed settings:

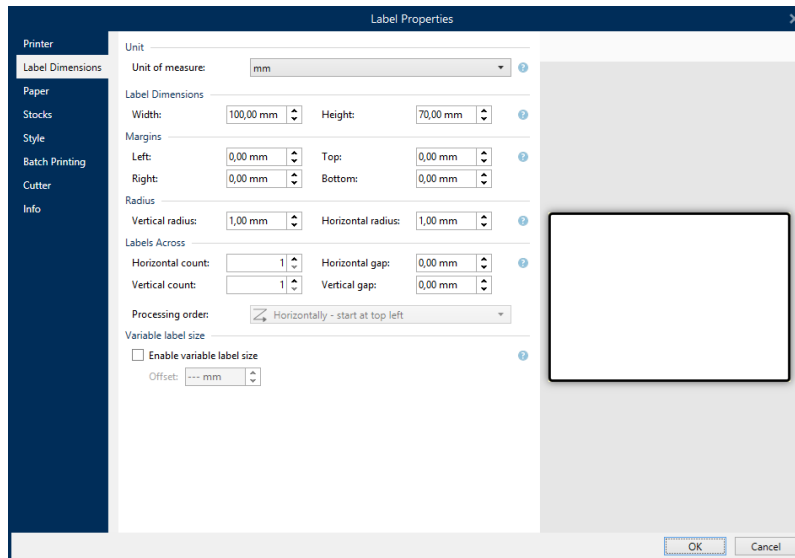
- **Printer:** selected printer for label printing.
- **Label dimensions:** dimensions of newly created label.
- **Paper dimensions:** dimensions of newly created label.

Label Properties

Label Properties dialog selects the printer, sets label dimensions and defines the printing paper properties.

The settings are available on the below listed dialog tabs.

Label Property	Description
Printer	Selects the preferred printer.
Label Dimensions	Defines the Unit of measure and label dimensions.
Paper	Defines the printing paper properties.
Stocks	Selects the stock type.
Style	Defines the label style parameters.
Batch Printing	Defines details for grouped printing of labels.
Cutter	Enables label roll cutting during or after the printing procedure.
Info	Inserts the label description.



TIP: To open the **Label Properties** dialog, double click the [design surface](#).

Printer

Printer tab lets you define the printer to print the labels on, and sets basic printing behavior.

Printer drop down menu selects a printer from the currently installed printers.

TIP: To set the printer settings, select a printer and click **Printer properties**. This button gives direct access to the selected printer's driver and its settings.

NOTE For additional information on the installed printer drivers and their settings, read the [NiceLabel Driver Installation Manual](#).

- **Always use the default printer:** selects the default system printer to be used for the current print job.
- **Double-sided printing:** enables double-sided label printing.
- **Use custom printer settings saved in the label:** each label may have its own printer settings defined and saved by the user. Select this option to use these settings while printing.
- **Use default printer settings from the printer driver:** select if you prefer the default printer settings or if no custom settings have been defined. Default printer settings are going to be used for printing.

Printing group of settings optimizes the printing process.

- **Optimize printing of identical labels:** if multiple identical labels are printed, the printer does not need to receive the label file each time. With this option enabled, the printer alone multiplies the print job.
- **Use advanced printer driver interface:** speeds up label printing.

TIP: When selected, the optimized printer commands are in use. Deselected option disables printing optimization. Each label is sent to the printer in form of an image.

- **Combine all non-printer elements into a single graphic item when sent to a printer:** merges multiple items and sends it to the printer as a single printable graphic.
- **Use store/recall printing mode:** optimizes printing performance.

With this mode activated, the Designer does not need to resend the complete label data for each printout. Instead, default labels (templates) are stored in the printer memory and the Designer only sends recall commands to complete the label content during the printing process. For more information, read section [Use Store/Recall Mode](#).

- **Store variant:** printer memory location to store the label templates.

NOTE To make sure the stored label samples are not lost after power cycling the printer, store them at non-volatile locations.

Label Dimensions

Label Dimensions tab specifies label dimensions and defines whether its size should adapt to the changing size of the objects or not.

Unit of measure defines the unit to be used while designing the label. There are four available units: cm, in, mm, and dot.

Label Dimensions group defines the label's **Width** and **Height**.

NOTE When manually inserting the unit of measure, this also changes the currently defined **Unit**.

Margins group sets the distance between the edge of the printing surface and the edge of the label (left/right, top/bottom).

Most laser and other non-thermal printers cannot print over the entire label surface. There is usually a non-printable label area of about 5 mm from the border of a page. In Designer, this area is marked by a red line. Any object on or beyond the red line is not printed entirely.

Radius group enables you to make the label corners rounded.

- **Vertical radius:** adjusts roundness value in vertical direction.
- **Horizontal radius:** adjusts roundness value in horizontal direction.

Labels Across defines the number of labels to be printed on a single label sheet.

- **Horizontal count:** labels distributed horizontally.
- **Vertical count:** labels distributed vertically.
- **Horizontal gap:** horizontal distance between labels on a sheet.
- **Vertical gap:** vertical distance between labels on a sheet.

Variable Label Size enables the label size to change in accordance with the size of its objects.

When assigning additional data to label objects, their size increases and occupies more space. Therefore, the label height must adapt.

- **Offset:** distance between the last object on a label and the bottom edge of a label.

Paper

Paper tab sets printing paper properties.

Unit selects the **Unit of measure** to be used in a label.

Paper Type group defines paper dimensioning type – automatic or manual.

- **Automatically set page size based on the label dimensions (labels on a roll):** page size is defined by the printer driver.

NOTE If a thermal printer is selected in the previous wizard step, this option is enabled by default.

- **Manually set page size (sheets of paper):** page size is set manually.

NOTE If a regular office laser printer is selected in the previous wizard step, this option is enabled by default.

In case the page size is defined manually, additional options appear:

- **Paper:** selection of standard paper formats.
- **Width and Height:** custom paper dimensions.

Orientation group sets the new label layout as **Portrait** or **Landscape**.

- **Rotated: Printer Layout** rotation for 180 degrees.

Preview displays current label screen and print layouts.

Stocks

Label stocks are a time-saving alternative to designing labels from scratch. Use stock templates when designing labels for a specific printer type and when optimizing the label designing process.

Stock type defines which stock type should be used when designing and printing a label. Stock types are usually associated with printer vendors or stationery suppliers.

NOTE Here defined stock properties override manually set label properties.

Stock defines the exact stock which belongs to the **Stock type**.

NOTE If the selected stock is not compatible with the selected printer, a warning appears. Label designing and printing becomes impossible.

Stock information displays the selected stock's properties:

- **Label dimensions**
- **Labels across**
- **Description**
- **Author**

Style

Style tab is used for defining label style parameters.

Background color is defined by the **Standard** or **Advanced** color selection.

Background picture sets a picture on the label background.

- **Embed the picture to document:** makes the picture an integral part of label document.
- **Save embedded picture to file:** embedded picture is saved to a file.
- **Picture position:** background picture to be centered, to fit the label dimensions, or to be stretched.
- **Rotation:** background picture rotation by 90 degrees.
- **Print background picture:** background picture is printed.

Batch Printing

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

Batch printing allows grouped printing of labels that belong to the same batch.

TIP: A batch is a set of labels printed within a single print job. Each print job can consist of a single or multiple batches.

The first purpose of batch printing is to automate the execution of a predefined action after the batch has been printed.

EXAMPLE Label roll is automatically cut after a batch of five labels has been printed.

The second purpose of batch printing is to enable header and tail label printing with each batch.

EXAMPLE A batch of five labels starts with a header and ends with a tail label. Both of them differ from the main (body) labels.

- **Enable batch printing:** activates batch printing mode. Batch definition menu becomes active.
- **Batch definition:** specifies what a batch of labels should consist of:
 - **All labels in the print job:** all labels in the current print job are assigned to the same batch.
 - **Batch ends after a specific number of labels:** batch is finalized after a specified number of labels is printed.
 - **Batch ends when the data source changes value:** changed value of the selected variable is used as a marker for opening a new batch.

Actions group defines the action that executes after a batch has been printed. The availability of actions depends on the selected printer's driver. If the driver provides no information on the action availability, the list is empty.

EXAMPLE Commonly used batch actions are **Cutter**, **Pause printer**, **Batch mark**, **Batch separator**, etc. With a defined web of labels (label template with labels next to each other), an applicable action also becomes **Eject page**. These printer commands can be applied dynamically during the printing process.

Header / Tail Labels group specifies the properties of header and tail labels in a batch.

- **Use header label:** header label of a batch.
- **Action after header label:** action to be taken after the header label has been printed. The selection of available actions depends on the selected printer's driver.

NOTE The selection of available actions depends on the selected printer's driver.

- **Use tail label:** last label of a batch.
- **Action after tail label:** action to be taken after the tail label has been printed.

NOTE The selection of available actions depends on the selected printer's driver.

TIP: Header, tail and main (body) labels of a single batch are accessible via tabs that are located under the design surface (gray field).

Cutter

Cutter enables label roll cutting during or after the printing procedure.

Enable cutter activates the label cutter and its settings.

Cutter mode specifies when the label roll is cut.

- **Cut after the last printed label:** label roll is cut after the finished print job.
- **Cut after a specific number of labels:** label roll is cut at a defined number of labels or if a defined condition is met.
- **Cut when the data source changes:** label roll is cut when the value of a data source changes or if a defined condition is met.

Info

Info tab includes a **Description** that serves as a hint or as a guidance for the user that is going to work with the label.

Define label **Description** by entering text into the field.

Label Objects

After setting the [label properties](#), it's time to start adding content to the label. Label objects are basic design items that are used for adding and editing various content types. Each object has its own function as described in the table below.

Label Object	Icon	Description
Text		Object for single-line textual content that does not need to adapt its font size to the label design.
Text box		Object which adapts (expand/shrink) to the textual content or makes the font increase or decrease its size to fit in the frame.
Rich text box		Object for adding textual content that supports formatted text, hyperlinks, line images, and other rich content created with a word processor.

Label Object	Icon	Description
Barcode		Object for adding and editing various types of barcodes on a label.
Picture		Object for adding graphic content to a label.
Rectangle		Object for creating rectangle shaped frames on a label
Line		Object for creating lines on a label.
Ellipse		Object for creating circular shapes on a label.
Inverse		Object for inverting the color of the underlying object.

Text

Text object encloses single-line textual content that does not adapt its font size to label design. Text object can also include multiple-line textual content – the object grows to fit the amount of entered text.

TIP: [Text box object](#) serves an alternative when designing a label on which the textual content must fit into a field with predefined dimensions.

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- [Variable keyboard input:](#) type of variable that enables the content of a prompted field to be different for every print job.
- [Variables:](#) predefined variable values which are used as object content.
- [Functions:](#) input data transformation tools.
- [Databases:](#) database values which are used as object content.
- [Link to other objects:](#) makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Content Mask sets the format of the input data before it is displayed on a label.

Mask character is a character used in the mask that is replaced with actual data on the printed label.

EXAMPLE

A user needs to format a phone number to be more readable on the label. Data input is not formatted since it is read from a database.

If the input value read from a database is:

+38642805090

and the content mask is:

(****) **** - ****

the resulting output is:

If the data contains the asterisk "*" character, change the **Mask character**. The character should have a unique value that does not appear anywhere in the data.

Style

Font color sets text font and underline color.

Font selects the typeface. Fonts are divided into two groups: OpenType fonts and Printer fonts.

NOTE If the currently selected printer is a thermal printer, additional fonts become available. These are the internal **Printer fonts** that are installed on the printer. Printer fonts are identified by the printer icon in front of their names.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Font Scaling sets the font stretch factor. If the factor is set to 100 %, font appears in normal scale. If the factor is set to 200 %, font appears twice as wide as normally. If set to 50 %, font is shrunk to half of its size.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.

Spacing sets the space between text characters and lines.

- **Line spacing:** space between each line in a paragraph.
- **Character spacing:** space between individual characters.

Effects

Inverse: inverted text and object background colors.

Mirror: mirrored text.

RTL printing: right to left text printing.

Most thermal printers print right-to-left scripts automatically. This option becomes useful if the operating system does not provide native RTL support.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object's position.



- **X** and **Y:** anchoring point coordinates.

Size group gives an information about the object's dimensions.

- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

In Text object, the size of text is determined by the font size. Object dimensions and aspect ratio cannot be changed manually and only serve as an information about its current size.

Rotation angle is the object angle according to the design surface.

There are multiple ways to set the object's angle: enter the angle manually , drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Text Box

Text box object is a fixed-size frame for adding and displaying textual content on a label.

Text box object is very similar to the standard Designer [Text](#) object. The difference between these two is the presentation of textual content with variable length. Text object is always expanding or shrinking to adapt to the amount of entered content. Text Box in opposite can either adapt (expand/shrink) to the content or make the font increase or decrease its size to fit in the object frame.

TIP: To make sure the content fits the predefined box is especially useful when working with variable data. No matter how long the text value is, it is always placed and displayed on a label within the predesigned frame.

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- [Variable keyboard input:](#) type of variable that enables the content of a prompted field to be different for every print job.
- [Variables:](#) predefined variable values which are used as object content.
- [Functions:](#) input data transformation tools.
- [Databases:](#) database values which are used as object content.
- [Link to other objects:](#) makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Mask sets the format of the input data before it is displayed on a label.

Content mask sets the format of the input data before it is displayed on a label.

Mask character is a character used in the mask that is replaced with actual data on the printed label.

E X A M P L E

A user needs to format a phone number to be more readable on the label. Data input is not formatted since it is read from a database.

If the input value read from a database is:

+38642805090

and the content mask is:

(****) **** - ****

the resulting output is:

(+386) 4280 - 5090

If the data contains the asterisk "*" character, change the **Mask character**. The character should have a unique value that does not appear anywhere in the data.

Style

Font color sets text font and underline color.

Font selects the typeface. Fonts are divided into two groups: OpenType fonts and Printer fonts.

NOTE If the currently selected printer is a thermal printer, additional fonts become available. These are the internal **Printer fonts** that are installed on the printer. Printer fonts are identified by the printer icon in front of their names.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Font Scaling sets the font stretch factor. If the factor is set to 100 %, font appears in normal scale. If the factor is set to 200 %, font appears twice as wide as normally. If set to 50 %, font is shrunk to half of its size.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.

Spacing sets the space between text characters and lines.

- **Line spacing:** space between each line in a paragraph.
- **Character spacing:** space between individual characters.

Text Fit

None makes Text box size and font non-adaptable.

NOTE If the content amount exceeds the object size, an error message appears. The label is not printed. To suppress such error and print the text box, enable **Ignore**.

Adjust height to fit content: automatic adaptation of Text box height.

Fit content by adjusting font size: increases or decreases the font size to make it fit inside the Text Box object.

- **Minimum size:** minimum permitted font size.
- **Maximum size:** maximum permitted font size.

Fit content by scaling font: shrinks or stretches the font to make it fit inside the Text Box object.

- **Minimum font scaling:** minimum font stretch factor.
- **Maximum font scaling:** maximum font stretch factor.

Effects

Inverse: inverted text and object background colors.

Mirror: mirrored text.

RTL printing: right to left text printing.

Most thermal printers print right-to-left scripts automatically. This option becomes useful if the operating system does not provide native RTL support.

Boundaries

Left border group defines the text boundary along the object's left border.

- **Shape:** selects a customizable basic shape of text boundary.
- **Width:** extends or shrinks the selected basic left boundary horizontally.
- **Height** extends or shrinks the selected basic left boundary vertically.

Right border group defines the text boundary along the object's right border.

- **Right shape** selects the basic shape of the object's right boundary.
- **Width** extends or shrinks the selected basic right boundary horizontally.
- **Height** extends or shrinks the selected basic right boundary vertically.

EXAMPLE Boundary defines how the text flows inside the object.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin aliquam id augue sed porttitor. Nunc sit amet dui justo. Aliquam condimentum mauris arcu, at hendrerit metus elementum eu. Morbi tristique libero ac turpis consequat, nec efficitur tortor malesuada.

Sed gravida odio at augue scelerisque aliquet.

Suspendisse imperdiet eget orci non bibendum. Aenean mattis nunc vitae pretium porttitor. Donec facilisis eleifend urna in vehicula.

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed, the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Rich Text Box

Rich text box (RTF) is an object for rich text editing. It encloses textual content with hyperlinks, line images, and other formatting created using an internal Designer's word processor.

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- [Variable keyboard input](#): type of variable that enables the content of a prompted field to be different for every print job.

- [Variables](#): predefined variable values which are used as object content.
- [Functions](#): input data transformation tools.
- [Databases](#): database values which are used as object content.
- [Link to other objects](#): makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Rich Text Box Editor is a full-scale text processor.

Edit content button opens the editor.

Supported actions in Rich Text Box Editor:

- Text formatting
- Content find and replace
- Inserting of images, symbols, tables, and dynamic data sources
- Content zooming

Show RTF code option displays the RTF code.

Read more about the available **Rich Text Box Editor** features [in a dedicated topic](#).

Text Fit

None makes Text box size and font non-adaptable.

- **None**: non-adaptable Rich Text box size and font.

NOTE If the content amount exceeds the object size, an error message appears. The label is not printed. To suppress such error and print the text box, enable **Ignore excessive content**

- **Adjust height to fit content**: automatic Rich Text box height adaptation.
- **Fit content by adjusting font size**: adaptable font size.
 - **Minimum size**: minimum font size.
 - **Maximum size**: maximum font size.

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y**: anchoring point coordinates.

Size group sets the object's dimensions:

- **Width** and **Height**: horizontal and vertical object dimension.
- **Keep aspect ratio**: simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Rich Text Box Editor

Rich Text Box Editor is a fully featured word processor. It enables you to create, edit and format the content of a **Rich text box** label object.

Sections below describe editor's tabs and matching ribbon groups with commands that are available when creating and editing the Rich Text box content.

Home Tab

File ribbon group enables handling of a document.

- **Import:** importing of textual content to the editor.
- **Export:** exporting of textual content from the editor.


Use file browser window to select the export location. By default, the editor content is exported as a file with .rtf extension. To specify an alternative file format, select it from the drop down list.

Clipboard ribbon group activates the following actions:


- **Paste:** pastes clipboard data.
- **Copy:** copies current selection to the clipboard.
- **Cut:** cuts selection to clipboard.

Undo Redo ribbon group undos or repeats editing actions.

Font ribbon group includes typical font style and formatting related commands. These are font selection, size, font growing and shrinking, bold, italics, etc.

For additional font related settings, open the **Font box** in dialog form by clicking the  icon in the bottom right corner of the ribbon group.

Text Box group defines lists and indents, toggles formatting symbols, sets alignment and line spacing, and enables text shading.

For additional text related settings, open the **Text Box** in dialog form by clicking the  icon in the bottom right corner of the ribbon group.

Editing group includes:

- **Find** searches and locates the inserted string within a text.
- **Replace** locates and replaces the inserted string with a new text.

Insert

Insert group enables adding editable elements to the rich text object.

- **Data Source:** adds variable, function or a database field as a dynamic content source.
- **Table:** opens the Insert Table dialog. Define **Number of columns** and **Number of rows**. After clicking **OK**, a table with the defined number of columns and rows is placed in the rich text editor.
- **Picture:** inserts a picture the rich text object.
- **Symbol:** opens **Insert Symbol** dialog for character selection.
 - **Search by code:** character search by unicode character code.
 - **Font name:** font selection.
 - **Character set:** active set of characters.
 - **Filter:** character search filter.

Frequently used symbols are displayed at the bottom of the dialog box. Click the symbol to insert it directly in the rich text object.

View

Zoom group allows zooming the text in and out.

Barcode

Barcode object is used for adding various types of barcodes on a label. Each barcode type is configurable according to specific standards.

TIP: When encoding the barcode content, make sure the used characters, length and identifiers comply with the barcode standard guidelines.

The following barcode types are available in Designer:

- [1D and 2D Barcodes](#)
- [GS1 DataBar Barcode Subtypes](#)

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- **Variable keyboard input:** type of variable that enables the content of a prompted field to be different for every print job.
- **Variables:** predefined variable values which are used as object content.
- **Functions:** input data transformation tools.
- **Databases:** database values which are used as object content.
- **Link to other objects:** makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Barcode

Barcode Type defines which barcode is used to encode the data.

TIP: Code128 barcode type is selected by default. For more details about the available barcode types, see section [Barcode Types and Available Settings](#).

- **X dimension:** width of the narrowest bar in the barcode.
- **Height:** barcode's Y dimension.
- **Ratio:** the ratio between **X dimension** and **Height**.

Each barcode type has the range of permitted ratios limited by the standard. Designer only permits using valid ratios.

Actual properties based on selected printer displays the X dimension as it would appear printed on a label using the currently selected printer.

Check Digit

Check digit is used by any scanning system to verify that the number scanned from a barcode is read correctly.

TIP: Check digit is derived from the preceding barcode digits and is placed as the final digit of a barcode.

Include check digit defines if check digit is included in a barcode or not.

- **Auto-generate check digit:** automatic check digit calculation.

NOTE If the data already includes invalid check digit, Designer replaces it with a proper value.

- **Verify the provided check digit:** verification of the manually provided check digit. An error message appears if the check digit proves to be incorrect.
- **Display in human readable:** check digit included in the human readable barcode text.

Human Readable

Human Readable text displays readable barcode data content located beneath or above the barcode. Its role is to provide backup in case the barcode is damaged or of

poor quality.

- **No human readable:** barcode is rendered without human readable text.
- **Above barcode:** human readable text above the barcode.
- **Below barcode:** human readable text below the barcode.

Mask sets the format of the input data before it is displayed on a label.

Content mask sets the format of the input data before it is displayed on a label.

Mask character is a character used in the mask that is replaced with actual data on the printed label.

E X A M P L E

A user needs to format a phone number to be more readable on the label. Data input is not formatted since it is read from a database.

If the input value read from a database is:

+38642805090

and the content mask is:

(****) **** - ****

the resulting output is:

(+386) 4280 - 5090

If the data contains the asterisk "*" character, change the **Mask character**. The character should have a unique value that does not appear anywhere in the data.

Bearer Bar

Bearer bar is a border that surrounds the barcode. Its purpose is to protect the barcode image and to enhance reading reliability.

- **Fixed thickness:** automatically defined bearer bar width.
- **Variable thickness:** user-defined bearer bar width.
- **Thickness multiplier:** bearer bar width factor.
- **Show vertical bar:** vertical bearer bars displayed or hidden.

Details

Details differ according to the barcode standards. Define the options that are given with regard to the currently selected barcode type. Details for 1D and 2D barcodes are described in dedicated sections:

- [1D barcode details](#)
- [2D barcode details](#)

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width** and **Height**: horizontal and vertical object dimension.
- **Keep aspect ratio**: simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on pre-designed or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Picture

Picture object is used for adding graphic content to a label. The following file formats are supported:

- Enhanced Windows Metafile (.emf)
- Windows Metafile (.wmf, .wmz, .emz)
- Portable Network Graphic (.png)
- TIFF bitmaps (.tiff)
- JPEG bitmaps (.jpg)
- PDF
- Windows bitmap (.bmp)

- Paint
- Adobe Photoshop

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- [Variable keyboard input:](#) type of variable that enables the content of a prompted field to be different for every print job.
- [Variables:](#) predefined variable values which are used as object content.
- [Functions:](#) input data transformation tools.
- [Databases:](#) database values which are used as object content.
- [Link to other objects:](#) makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

To (re)define the Picture object **Content**, click **Browse** and locate the file to be displayed on the label.

Embed picture in a document stores the picture in the label file. Link to the original picture file is discarded.

TIP: Picture embedding makes the label file more portable as the user does not have to re-include the picture file in case of repeated use.

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed, the value converts automatically.

Graphic Resizing

Graphic Resizing is available if the picture object is connected to a variable. These settings define how the Picture object adapts its size to the source file at print time.

- **Keep original picture size:** disabled picture resizing. Picture size remains unchanged.
- **Resize proportionally:** proportional picture resizing. Aspect ratio of picture dimension remains fixed.
- **Resize to the designed size:** horizontal and vertical picture resizing to make it fit into the bounding box. This option will most likely make the picture distorted.

Original size displays the picture's **Width** and **Height** before resizing. **Revert to original picture size** undos the resizing actions.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Rectangle

Rectangle object creates a rectangle shaped frame on a label.

Style

Outline group defines line settings:

- **Thickness:** object line thickness.
- **Line style:** object line style:
 - **None:** line invisible.
 - **Solid:** solid line.
 - **Dot:** dotted line.
 - **Dash:** dashed line.
 - **Erase:** parts of neighboring objects become invisible underneath the Rectangle line.
- **Line color:** color of the line.
- **Corner radius:** makes the rectangle corners rounded.

Fill defines the object fill settings and color.

- **Fill style:** object fill properties definition:
 - **None:** completely transparent object.
 - **Erase:** invisible objects beneath the active one.
 - **Solid:** fills the object with solid color.
 - **Right Diagonal:** fills the object with diagonal lines that ascend toward the right side.
 - **Left Diagonal:** fills the object with diagonal lines that ascend toward the left side.
 - **Vertical:** fills the object with vertical lines.
 - **Horizontal:** fills the object with horizontal lines.
 - **Cross:** fills the object with crossed lines.
 - **Cross Diagonal:** fills the object with diagonally crossed lines.
 - **25% of color:** fill color opacity 25 %.
 - **50% of color:** fill color opacity 50 %.
 - **75% of color:** fill color opacity 75 %.
- **Fill color:** object fill color definition.

NOTE The system does not allow the **Line style** and **Fill style** to be set to **None** at the same time.

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X** and **Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually , drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed, the value converts automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.

- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Line

Line object is used to create a line on a label.

Style

Outline group defines line settings:

- **Thickness:** object line thickness.
- **Line style:** object line style:
 - **None:** line invisible.
 - **Solid:** solid line.
 - **Dot:** dotted line.
 - **Dash:** dashed line.
 - **Erase:** parts of neighboring objects become invisible underneath the Rect-angle line.
- **Line color:** color of the line.

Position

Position tab defines object positioning and its position-related behavior.


Position group defines the object's position.


- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually , drag the slider or click and drag the  icon on the selected object. Rotation angle and

slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Ellipse

Ellipse object is used for creating a circular shaped object on a label.

Style

Outline group defines line settings:

- **Thickness:** object line thickness.
- **Line style:** object line style:
 - **None:** line invisible.
 - **Solid:** solid line.
 - **Dot:** dotted line.
 - **Dash:** dashed line.
 - **Erase:** parts of neighboring objects become invisible underneath the Rectangle line.
- **Line color:** color of the line.

Fill defines the object fill settings and color.

- **Fill style:** object fill properties definition:
 - **None:** completely transparent object.
 - **Erase:**invisible objects beneath the active one.
 - **Solid:** fills the object with solid color.
 - **Right Diagonal:** fills the object with diagonal lines that ascend toward the right side.
 - **Left Diagonal:** fills the object with diagonal lines that ascend toward the left side.
 - **Vertical:** fills the object with vertical lines.
 - **Horizontal:** fills the object with horizontal lines.
 - **Cross:** fills the object with crossed lines.
 - **Cross Diagonal:** fills the object with diagonally crossed lines.
 - **25% of color:** fill color opacity 25 %.
 - **50% of color:** fill color opacity 50 %.
 - **75% of color:** fill color opacity 75 %.
- **Fill color:** object fill color definition.

NOTE The system does not allow the **Line style** and **Fill style** to be set to **None** at the same time.

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed, the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Inverse

About

Inverse object inverts the underlying object's color .



Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width** and **Height**: horizontal and vertical object dimension.
- **Keep aspect ratio**: simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
 - **Border:** reference label or neighboring object's border for horizontal relative positioning.
 - **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed, the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

Selecting And Setting Up A Printer

Printer

Printer tab lets you define the printer to print the labels on, and sets basic printing behavior.

Printer drop down menu selects a printer from the currently installed printers.

TIP: To set the printer settings, select a printer and click **Printer properties**. This button gives direct access to the selected printer's driver and its settings.

NOTE For additional information on the installed printer drivers and their settings, read the [NiceLabel Driver Installation Manual](#).

- **Always use the default printer:** selects the default system printer to be used for the current print job.
- **Double-sided printing:** enables double-sided label printing.

- **Use custom printer settings saved in the label:** each label may have its own printer settings defined and saved by the user. Select this option to use these settings while printing.
- **Use default printer settings from the printer driver:** select if you prefer the default printer settings or if no custom settings have been defined. Default printer settings are going to be used for printing.

Printing group of settings optimizes the printing process.

- **Optimize printing of identical labels:** if multiple identical labels are printed, the printer does not need to receive the label file each time. With this option enabled, the printer alone multiplies the print job.
- **Use advanced printer driver interface:** speeds up label printing.

TIP: When selected, the optimized printer commands are in use. Deselected option disables printing optimization. Each label is sent to the printer in form of an image.

- **Combine all non-printer elements into a single graphic item when sent to a printer:** merges multiple items and sends it to the printer as a single printable graphic.
- **Use store/recall printing mode:** optimizes printing performance.

With this mode activated, the Designer does not need to resend the complete label data for each printout. Instead, default labels (templates) are stored in the printer memory and the Designer only sends recall commands to complete the label content during the printing process. For more information, read section [Use Store/Recall Mode](#).

- **Store variant:** printer memory location to store the label templates.

NOTE To make sure the stored label samples are not lost after power cycling the printer, store them at non-volatile locations.

Working With Objects

This section describes how to work with [objects](#) to make them blend with the design of a [label](#) or [form](#).

Object is a basic building block of any label or solution. Each object is dedicated to a specific type of content. See the related topics for style and content related object properties.



The below listed actions describe which actions are common for multiple object types:

- **Adding an object:** add an object to a label or form, select the appropriate one in the object toolbar and drag it to the design surface.
- **Adding an object with connected data source:** click the down arrow next to the object button and select an existing or new data source to make the newly added object instantly connected to a dynamic data source.

- **Grouping:** make multiple object behave as a single object. Hold <SHIFT> and click the objects select multiple object and click Group objects to create a group of objects.
- **Rotating:** Changes the angle of a selected object. Details on how to rotate the objects are available [here](#).
- **Resizing:** Sets the size of an object.
- **Aligning:** make the object positions [aligned](#).

Object Rotating

There are two ways to set the angle of an object:

- Enter the angle manually in degrees or drag the slider. The object rotates around its anchoring point. Rotation commands are accessible in two ways:
 - Click **Position** in the [Positioning group](#) of the Design tab
 - Go to **Object properties -> Position -> Rotation angle**.
- Click and drag the  icon next to the selected object. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Barcode

Barcode object is used for adding various types of barcodes on a label. Each barcode type is configurable according to specific standards.

TIP: When encoding the barcode content, make sure the used characters, length and identifiers comply with the barcode standard guidelines.

The following barcode types are available in Designer:

- [1D and 2D Barcodes](#)
- [GS1 DataBar Barcode Subtypes](#)

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- [Variable keyboard input:](#) type of variable that enables the content of a prompted field to be different for every print job.
- [Variables:](#) predefined variable values which are used as object content.
- [Functions:](#) input data transformation tools.
- [Databases:](#) database values which are used as object content.
- [Link to other objects:](#) makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Barcode

Barcode Type defines which barcode is used to encode the data.

TIP: Code128 barcode type is selected by default. For more details about the available barcode types, see section [Barcode Types and Available Settings](#).

- **X dimension:** width of the narrowest bar in the barcode.
- **Height:** barcode's Y dimension.
- **Ratio:** the ratio between **X dimension** and **Height**.

Each barcode type has the range of permitted ratios limited by the standard. Designer only permits using valid ratios.

Actual properties based on selected printer displays the X dimension as it would appear printed on a label using the currently selected printer.

Check Digit

Check digit is used by any scanning system to verify that the number scanned from a barcode is read correctly.

TIP: Check digit is derived from the preceding barcode digits and is placed as the final digit of a barcode.

Include check digit defines if check digit is included in a barcode or not.

- **Auto-generate check digit:** automatic check digit calculation.

NOTE If the data already includes invalid check digit, Designer replaces it with a proper value.

- **Verify the provided check digit:** verification of the manually provided check digit. An error message appears if the check digit proves to be incorrect.
- **Display in human readable:** check digit included in the human readable barcode text.

Human Readable

Human Readable text displays readable barcode data content located beneath or above the barcode. Its role is to provide backup in case the barcode is damaged or of poor quality.

- **No human readable:** barcode is rendered without human readable text.
- **Above barcode:** human readable text above the barcode.
- **Below barcode:** human readable text below the barcode.

Mask sets the format of the input data before it is displayed on a label.

Content mask sets the format of the input data before it is displayed on a label.

Mask character is a character used in the mask that is replaced with actual data on the printed label.

EXAMPLE

A user needs to format a phone number to be more readable on the label. Data input is not formatted since it is read from a database.

If the input value read from a database is:

+38642805090

and the content mask is:

(****) **** - ****

the resulting output is:

(+386) 4280 - 5090

If the data contains the asterisk "*" character, change the **Mask character**. The character should have a unique value that does not appear anywhere in the data.

Bearer Bar

Bearer bar is a border that surrounds the barcode. Its purpose is to protect the barcode image and to enhance reading reliability.

- **Fixed thickness:** automatically defined bearer bar width.
- **Variable thickness:** user-defined bearer bar width.
- **Thickness multiplier:** bearer bar width factor.
- **Show vertical bar:** vertical bearer bars displayed or hidden.

Details

Details differ according to the barcode standards. Define the options that are given with regard to the currently selected barcode type. Details for 1D and 2D barcodes are described in dedicated sections:

- [1D barcode details](#)
- [2D barcode details](#)

Position

Position tab defines object positioning and its position-related behavior.



Position group defines the object's position.

- **X and Y:** anchoring point coordinates.

Size group sets the object's dimensions:

- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.

Rotation angle is the object angle according to the design surface.

TIP: There are multiple ways to set the object's angle: enter the angle manually, drag the slider or click and drag the  icon on the selected object. Rotation angle and slider rotates the object around its anchoring point. The  icon rotates the object around its central point.

Anchoring point is the spot where an object is pinned to design surface. Variable size objects increase or decrease their size in the direction that is opposite to the chosen anchoring point.

Lock object on design surface prevents the object from being moved during the design process, select under the **Design behavior** group.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

Relative Position

Relative Position defines object position when the label size or neighboring object's positions are changing during the design process.

- **Enable horizontal relative position:** activates horizontal relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its horizontal offset according to this border.

- **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its horizontal offset according to this object.
- **Border:** reference label or neighboring object's border for horizontal relative positioning.
- **Offset:** horizontal distance from label border or selected object's anchoring point.
- **Enable vertical relative position:** activates vertical relative positioning.
 - **Relative to label border:** the position of object is defined relative to the reference label border. Define its vertical offset according to this border.
 - **Relative to another object:** the position of object is defined relative to the selected neighboring object's border. Define its vertical offset according to this object.
 - **Border:** reference label or neighboring object's border for vertical relative positioning.
 - **Offset:** vertical distance from label border or selected object's anchoring point.

NOTE Object position changes if label size or position of the related object are changed.

NOTE If the measurement unit is changed in [label properties](#), the value transforms automatically.

General

General tab identifies the object and sets its status.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

NOTE NiceLabel recommends avoiding spaces or special characters in object names.

Description allows adding notes and annotations for an object. It provides help during the label design process.

Status group defines object visibility on print preview and on printed labels.

- **Not printable:** prevents the object from being printed. The object still remains visible on the print preview and affects other objects in relative positioning. This option is useful when printing on predesigned or stock-specific labels.
- **Visible:** if the check box is not selected, the object neither appears on the print preview nor on the printed label. The object is treated as if it does not exist at all.
- **Condition:** makes an object enabled (editable) if the result of the given condition is "True". It defines object visibility on form startup and when the connected variable's value changes.
- Object visibility status is explained in the table below:

Option	Print Preview	Printout	Relative positioning
Not printable (selected)	YES	NO	YES
Visible (cleared)	NO	NO	NO

Printing optimization group allows activating the use of printer elements.

- **Use printer elements if supported** speeds up the printing process.

If enabled by the printer model, a share of label element processing is handled directly by the printer: internal fonts, shapes, barcodes, etc.

- **Always print as graphics** sends and prints the objects as graphic files.

1D Barcode Details

Details tab settings vary along with the specific barcode standards.

TIP: Define the available barcode settings with regard to the currently selected barcode type.

Designer allows setting the following 1D barcode details:

- **Include quiet zones:** blank space around the printed barcode. Quiet zone ensures the highest level of scanning reliability.
- **Inter character gap:** distance between the last bar of a character and the first bar of the next character in a barcode.
- **Descender bars:** makes the bars at the beginning, in the middle, and at the end of certain barcode types (EAN and UPC) longer.
- **Include EAN white space:** inserts a special character ("<" or ">") to indicate the EAN barcode width.

This option ensures optimum readability in case a neighboring object on a label is located right next to the barcode.

- **Space correction:** adds white pixels to increase the gap width between the bars.
- **Symbology:** UPC barcode **Number system:**
 - 0, 1, 6, 7 and 8 are for regular UPC codes.
 - 2 is for random weight items, e.g. meat, marked in-store.
 - 3 is for National Drug Code and National Health related Items.
 - 4 is for in-store marking of non-food items.
 - 5 and 9 are for coupon use.

2D Barcode Details

2D barcodes enable multiple type-specific settings under the **Details** tab. When defining these settings manually, the drop down lists offers specific standard-compliant options.

TIP: Designer defines the **Details** tab settings automatically if the user chooses not to manually define them.

Code Page

Code page defines how the mapping of code characters with scanned characters is done. To display the scanned data accurately, the correct code page must be selected. If

none of the code pages is selected by the user, Designer uses system character encoding.

Columns

Columns are basic vertical elements of a PDF 417 barcode. A maximum of 30 columns may be included in a single PDF 417 symbol. Each column is 10 modules wide, which means a single barcode is capable of encoding up to 929 symbol characters. Theoretically, a single PDF417 barcode can store up to 1850 alphanumeric characters, 2710 digits or 1108 bytes.

Compaction Mode

Compaction mode compacts a number of data characters into codewords. The decoding algorithm uses the individual codewords to place them into a meaningful matrix.

- **Text:** all printable ASCII characters 32–126 and ASCII 9, 10 and 13 (up to 1800 characters) are allowed.
- **Binary:** all 256 ASCII values (up to 1100 bytes) are allowed.
- **Numeric:** encoding of numeric data (up to 2700 digits).

Data Layer

Data layer defines the number of data layers that encode data in an Aztec barcode. The number of data layers correlates directly with the barcode data capacity. If the value exceeds the data capacity provided by the selected Data layer, an error is reported. 1 to 4 data layers are allowed.

Encoding

Encoding defines character encoding scheme for the active barcode.

Error Correction Level

Error correction level defines the symbol security level. It adds a series of error correction codewords to the encoded data. These codewords enable the printed symbol to withstand damage without data loss. The higher the security level, the greater the number of data layers required to contain the symbol – and hence its overall size. If none of the Error correction levels is selected, Designer defines it automatically.

Format

Format defines the symbol size and its capacity using the number of column and row elements.

Rows

Rows – PDF417 barcode symbol is made of stacks of vertically aligned rows. Such barcode adapts its size to the amount of the encoded data and may contain from 3 to 90 rows.

Symbol Version

Symbol version defines the symbol data capacity. As the amount of data increases, additional modules are required to build a QR code. This makes the symbol larger on the printed label.

Truncated

Truncated reduces the PDF417 barcode size by removing a single codeword and a stop bar from each symbol row.

Version

Version defines the symbol size based on the number of columns. One-, two-, three-, and four-column version of Micro PDF417 barcode are available.

GS1 DataBar Specifics

in addition to the [common barcode properties](#), the below described specifics are available for GS1 DataBar.

GS1 DataBar Source

General groups specifies how the databar content is going to be formatted before encoding.

- **Structured data** sets the standard GS1 system data structure as a model for inserting the barcode data. Use [GS1 function](#) to encode the data correctly (for more on GS1 and other functions, see [topic Functions](#)). Composite GS1 barcodes represent structured data in the composite part of the code.
- **Unstructured data** allows inserting the data without a model – only character type and number must comply with the selected barcode type.

Data

- **Linear data** is the part of the data that is encoded in the linear part of the barcode. The data is either manually inserted or defined by a predefined **Data source**.
- **Composite data** is the part of the data that is encoded in the composite part of the barcode. This part of data is always structured and follows one of the standard system data structures as defined by the GS1. The data is either manually inserted or defined by a predefined **Data source**.

GS1 DataBar Properties

GS1 DataBar Expanded Stacked subtype encodes the data in form of a symbol segments sequence. Symbol width is defined by the number of symbol segments in each stacked row. Symbol height is defined by the number of stacked rows and their height.

- **Segments per Row** defines the number of segments for each row of a symbol. Up to 22 segments are allowed per symbol. A higher number makes the symbol longer. A lower number increases the symbol in height.

Maxicode Barcode Content

Symbology Definition defines the barcode mode of operation (data structuring type).

Designer supports the following modes:

- **Mode 2:** US carriers with postal codes up to 9 digits in length.
 - **Postal Code:** US Zip Codes using a single field with 5 or 9 digits, or two fields with 4 or 5 digits.
- **Mode 3:** international carrier with alpha-numeric postal codes with up to 6 digits.

There are two additional options under **Symbology Definition**:

- **Structured data:** automatically selected **Mode 2** or **Mode 3** modes based on the entered data.
- **Unstructured data:** barcode mode of operation is set to **Mode 4**.

This mode encodes general data for purposes other than shipping industry (e.g. purchase order number, customer reference, invoice number).

Data Contents

SHIP TO Postal Code	Mandatory. 5 or 9 alphanumeric characters. Alpha characters must be upper case.
4 Digit Extension (enabled with Postal code field Two Fields (5 and 4 digits) type).	Mandatory. 4 numeric digits defining micro location.
SHIP TO ISO Country Code (Mode 3 only)	Mandatory. 3 numeric digits.
Class of Service	Mandatory. 3 numeric digits, a comma must be included to mark the end of field.
Transportation Data	Mandatory. The 5 characters, including the GS code.
Tracking number	Mandatory. 10 or 11 alphanumeric characters. Alpha characters must be upper case.
UPS SCAC	Mandatory. 4 characters followed by the GS code.
Julian Day of Pickup	Mandatory. 3 numeric digits.
Shipment ID Number	Optional. 0-30 alphanumeric characters. Alpha characters must be upper case. GS code must always be sent even if no data is specified.
Package in Shipment	Mandatory. 1-3 numeric digits for package number. 1-3 numeric digits for number of shipped items. Forward slash must separate these two numbers.
Package in Weight	Mandatory. 1-3 numeric digits.
Address Validation	Mandatory. Single character "Y" or "N". Upper case characters.
SHIP TO Address	Optional. 0-35 alphanumeric characters. Alpha characters in upper case. GS code must always be sent even if no data is specified.
SHIP TO City	Mandatory. 1-20 alphanumeric characters. Alpha characters must be upper case.
SHIP TO State	Mandatory. 2 alpha characters. Both characters must be upper case. RS code marks the end of this field and the end of the secondary message data.

Printing

PRODUCT LEVEL INFO Solution building is available in NiceLabel PowerForms.

Use the below listed steps to successfully print a label using the NiceLabel Designer.

Step 1: Create

Create a new or edit an existing standalone [label](#) or label in a [solution](#).

Step 2: Preview

Label preview field is a part of default Designer [Print dialog](#). To make the print form appear on screen, select one of the following options:

- Go to [Home tab -> Action group](#) and click **Print**.
- Press **Ctrl+P**.

Label preview field displays the current label design. If you decide to customize the default print form or to make a new one, make sure you add the [Label Preview](#) object to the form. The print form will offer label preview only if the Label Preview object is present.

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

TIP: The default printing form is customizable. To adapt it and to create a custom print dialog, go to Home tab -> Action group and click **Customize Printing Form**. Read more about printing form customization [here](#).

Step 3: Select Printer

Choose the preferred printer from the **Printer** tab drop down menu. All currently installed printers are listed. More details on defining the printer are available [here](#).

During this step, printing speed and darkness can be set as well. These two parameters depend on the selected printer's driver.

Step 4: Set Print Quantity

Number of labels sets the number of printed labels.

Number of pages sets the number of printed pages. This option becomes active if the labels are positioned across at least two pages.

Print all labels (unlimited) prints all labels as defined by the label design. More details about this option are available [here](#).

Click **more...** to open the Additional Quantity Settings dialog.

- **Identical copies per label** defines the number of identical label copies in a print job.
- **Number of label sets** defines how many times the entire label printing process should repeat.

Step 5. Start Printing

Click the **Print** button.

Preview And Print A Label

PRODUCT LEVEL INFO Print form editing is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

When the label is ready to be printed, provides a print dialog which allows you to:

- preview the label before printing
- enter values for prompted variables variable keyboard input.
- filter and select which records should be printed if the label contains databases
- control printer settings
- control label print quantity
- set additional quantity settings

To open the print dialog, click the **Print** button in the [Action group](#) of the [Home tab](#) ribbon or press Ctrl+P.

The Designer print dialog is a customizable printing form. It consists of predefined form objects that can be configured, moved, added or removed. More details on how to use the printing form is available [here](#).

Customize Printing Form

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

In Designer, the [default printing form](#) serves as the basic print dialog for standalone and solution labels.

To start customizing the default printing form, go to **Home tab -> Action group -> Print** and click the down arrow button. Click **Customize Printing Form** to add, move, remove, and edit the printing form objects and their properties.

The variables that define the content of default print form objects become visible and editable in the [Solution Explorer](#). Open the [Dynamic Data Manager](#) to manage these variables.

If you want to discard the printing form customization actions, click **Recreate Printing Form**. The default printing form is recreated.

NOTE **Recreate Printing Form** discards all editing actions on the default printing form.

Printing Using NiceLabel Print

To print a label using the Print, complete the following steps.

1. Use **Manage Locations** dialog to select the location that stores the documents to be printed or run. See section [Managing Label Locations](#) for more details. All documents become visible and instantly printable in the document display area.

Skip this step in case of repeated printing from the same folder.

2. Select a document. Document type determines the actions that follows:
 - In case of a label file, click **Print** to open the printing form.
 - When opening a label file, [printing form](#) appears. It allows the user to preview the label, select the printer, define quantity settings, and enter prompted variable values (if included on the label).
 - To return to the label display field, click back arrow in the upper left corner of the Print window.
 - In case of a solution, click **Run** to open the solution in a separate instance.

TIP: All supported document types in the document display area are presented with previews for easier recognition and selection.

Read more about Print [here](#).

Store/Recall Printing Mode

Store/Recall printing mode optimizes the printing process. It increases printer response by reducing the amount of data that needs to be sent during repetitive printing tasks.

With store/recall mode activated, Designer does not need to resend the complete label data for each printout. Instead, default labels (templates) are stored in the printer memory and the Designer only sends recall commands which complete the stored label content during the printing process. Typically, a few bytes of data are sent to the printer, compared to a few kilobytes as would be the case during normal printing.

The action consists of two processes:

- **Store label.** During this process, Designer creates a description of the label template formatted in the selected printer's command language. When done, Designer sends the created command file to the printer memory and stores it.
- **Recall label.** A label stored in the printer memory is printed out immediately. Using the recall process, Designer creates another command file to instruct the printer which label from its memory should be printed. The recall label command occupies a few bytes of data only. The actual amount of data depends on the current situation. For fixed labels without any variable contents, the recall command file only contains the recall label command. For variable labels that contain variable fields, the command file includes the values for these variables and the recall label command.

NOTE Before activating this mode, make sure the appropriate printer driver is selected for the label printer. Not all label printers have the ability to use the store/recall printing mode.

Follow these steps to activate the **Store/Recall** printing mode:

1. Double click the label design surface. **Label Properties** dialog appears.
2. To enable the mode, select **Use store/recall printing mode** on **Printer** tab. Click **OK**.
3. Define the label template(s). All label objects with variable content must be formatted as internal printer objects:
 - Format the text object with internal printer fonts (not Truetype!).
 - Format barcode objects as internal printer barcodes.
 - If using variable objects formatted in Truetype fonts, variable pictures or database fields, default values are sent to the printer during the label store process.
4. Click **File -> Store**. Make sure the **Store variant** points to the correct memory location in the printer.
5. Insert or select the values for variable objects that are not formatted as internal printer objects. These variables will be given the same value on each label. They will behave as objects with fixed values.
6. Click **Store to printer** to create the command file with label template description and to send it to the printer.
7. Insert the values for prompted label variables. These variables link to the internal printer objects on the label. For this reason, their values can be changed during each printing.
8. Click **Print** to send the variable values and the recall label command to the selected label printer.

Optimize Printing Speed

There are many factors that affect the speed of label printing in Designer. Follow the guidelines below to dramatically increase the speed of printing.

NOTE When implementing the below listed guidelines, check if they are supported by the selected printer.

- If the selected printer supports parallel and serial port, use the parallel port. Computer sends the data to printer over parallel port much faster than over serial port.
- When designing a label, use internal printer fonts instead of Windows true-type fonts. True-type fonts are sent to the printer as graphics. This vastly increases the size of data sent to printer (couple of kilobytes). With internal printer fonts, only the text is sent to printer (couple of bytes).
- Avoid using graphics on labels.
- When using barcodes, make sure the barcodes are used as internal printer elements.

- When using counters, the printer internally increments the numbers if the internal printer fonts are used. This means, that the printer only needs to receive the first object number. The printer later increments this number while printing additional labels. This option also reduces the amount of data transferred between computer and printer.

With internal printer counter, the printing speed difference becomes noticeable with high quantity of labels.

- Set the printing speed to a higher value. Increasing the printing speed usually affects the quality of printing. The higher the speed, the lower the quality. Find an acceptable compromise.
- Don't print excessive amount of data on labels. If the speed of printing is an important factor, consider using preprinted labels, and only print the data, that changes with each label.

Handle Missing Images

Designer remembers the path and file name of pictures on a label. Each time a label is opened, Designer checks if the pictures are accessible and uses them on the label. If the picture is missing, a warning dialog box appears. The following options are available if the picture file is missing:

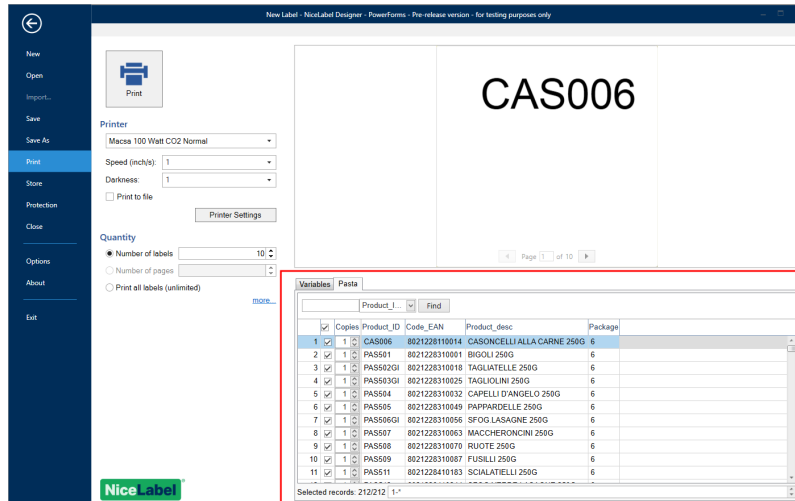
- Ignore the error and temporarily design the label without the missing picture.
- Discard the missing picture and permanently remove it from the label.
- In case the picture file location or name has changed, browse for the file and reselect it.

Printing From Databases

This section describes how to print the content of database records individually or in groups.

After completing the [database wizard](#), by default, all database records are printed. Each record is printed once per label.

If you do not want to print the entire database, select which records should be printed. Prior to printing, you the print dialog shows all database records. Use the data initialization field to select the records to be printed.



If you would like to print several copies of a label with record data, define this using:

- [Additional Quantity Settings dialog](#): Use **Identical copies per label** to set the desired quantity for the entire range of database records.
- Increase or decrease the value in **Copies** field of the data initialization field to set the number of printed labels per record individually.
- Use **Label copies per record** step of database wizard to dynamically define the number of printed labels per record.

EXAMPLE You have a database containing records with your products. It contains a field with a numeric value. This value stores the required number of label copies is stored. Select this field and let the application print the quantity of labels as specified in this field.

Changing Common Printer Settings

When designing a label, you also define which printer should be used by Designer to print it. Each label file remembers printer settings for the selected printer driver on the label.

Any changes made in the printer settings dialog box are saved to the label and applied to future print actions.

NOTE Make sure **Use custom printer settings saved in the label option** is enabled in **Label properties > Printer**. If not, default printer settings are going to be used.

Complete these steps to change and save common printer settings for a label:

1. Open [label properties](#) dialog.
2. Click **Printer properties** button on **Printer** tab. The dialog window with printer driver settings opens.
3. Open the **Printer Options** tab.

4. Adjust the **Speed** and **Darkness** settings.

Print settings		
Speed:	102 mm/s	<input checked="" type="checkbox"/>
Darkness:	3	<input checked="" type="checkbox"/>
Darkness range:	N/A	<input checked="" type="checkbox"/>

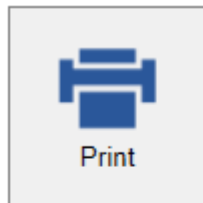
NOTE These settings depend on the selected printer.

5. Click **OK**.
6. Save the label.

Changes in label printing speed and darkness can also be done at print time. These settings are only valid until the file remains open. After reopening the file, the settings are again reset to those defined in **Printer properties** dialog.

Complete the following steps:

1. Open [Print dialog](#).
2. Click **Print**.
3. Adjust **Speed** and **Darkness** values under **Printer** group.
4. Save the label.



Printer

[Printer Name]	
Speed (inch/s):	1
Darkness:	1
<input type="checkbox"/> Print to file	
Printer Settings	

Changing Dithering Options

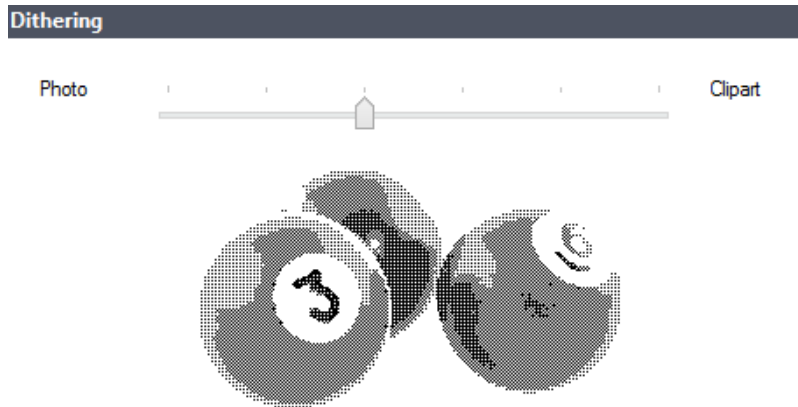
Dithering is a process of converting color or gray scale pictures to black and white pictures that can be printed on thermal printers. Thermal printers normally cannot print color images and can either print a dot on the label or leave the area blank. There are no intermediate shades of gray.

During the dithering process, all colors and shades of gray in the picture are converted to black and white dots, creating an illusion of new colors and shades by varying the

pattern of dots. Different shades of gray are produced by varying the patterns of black and white dots. There are no gray dots at all. In printing, dithering is usually called halftoning, and shades of gray are called halftones.

To change the dithering settings, do the following:

1. Open [label properties](#) dialog.
2. Click **Printer properties** button on **Printer** tab. The dialog window with printer driver settings opens.
3. Open **Graphic Options** tab and use **Photo** slider to select the preferred dithering type.



NOTE These settings depend on the selected printer.

4. Change the dithering type option to suit your needs. Look at the preview on the right side how you can expect the selected type to be applied on the label.
5. Click **OK**.
6. Save the label.

Defining Unprintable Area

Unprintable area is the part of the label where the printer cannot print. You can virtually increase the size of the label by enabling unprintable area in the printer driver.

Thermal printer can only print labels that are placed below the print head. If you have wider labels and if the print head does not completely cover the label, the label part which juts out of the print head cannot be printed.

With unprintable area feature you inform the Designer that there is an unusually wide label inserted into the printer. The software will draw vertical red lines identifying the unprintable area.

The unprintable area is usually the label area left and right of the printer head.

TIP: The unprintable area is not a margin. The label objects are not shifted on the label.

To define the unprintable area, do the following:

1. Open [label properties](#) dialog.
2. Click **Printer properties** button on **Printer** tab. The dialog window with printer driver settings opens.

3. Go to **Printer options** tab.
4. Enter the values for **Unprintable Area**.

EXAMPLE You have a printer with 10 cm (4") printer head and a 12 cm wide label. You insert the label centrally in the printer, so it sticks out of the print head evenly on both sides. You define a new label in the labeling software with 12 cm width. By setting the unprintable area to 1 cm on the left and 1 cm on the right side you let the labeling software know that the actual label width is 10 cm. There will be two vertical red lines on the design surface identifying the unprintable area.

NOTE Vertical red lines are also visible when you switch the printer for the same label. The original printer might had wider print head than the new printer. Maximum widths of the labels are not the same for both printers. The labeling software will try to preserve the original label dimension and automatically define the unprintable area for the new printer.

Work with Dynamic Data Sources

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Dynamic data sources form an essential part of working with the NiceLabel Designer. They enable the use of label and form objects that dynamically change their content with each printed label if necessary.

EXAMPLE Typical dynamic content examples that need to be automatically updated are counters, serial numbers, date, time, weight, and article images.

To display and print the dynamic object content properly, Designer uses the following dynamic data types:

- [Variable keyboard input](#): content of an object is defined before each printing.
- **Link to other object**: content of an object is defined by the content of another (linked) object on a label.
- [Variables](#): display and store dynamic data source values which are defined at print time.
- [Functions](#): transform the dynamic data source values. Functions define the output format to adapt the input–output conversion to specific requirements.
- [Databases](#): retrieve and display the database record .
- [Internal Variables](#): display the values that are automatically retrieved from a running application and system environment.
- [Global Variables](#): a type of variable that can be shared among multiple labels.

TIP: Read about how to navigate and manage dynamic data sources in topic [Dynamic Data Manager](#).

TIP: Read more about the relation between dynamic data sources and label/form objects in section [Dynamic Object Content](#).

Variables As Dynamic Data Source

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Variables serve as containers for storing and passing data between objects, scripts, external applications, printers, and user inputs. You may want to print labels on which data changes for each label. For example, counters, serial numbers, date and time, weight, article pictures... To accommodate the changing data, the labeling application can easily be used to format labels using variable data.

Designer offers multiple types of variables:

- **Variable:** type of variable that changes its value at print time or according to user-defined conditions.
- **Variable Keyboard Input:** type of variable that enables the content of a prompted field to be different for every print job. Its value is defined before each printing.
- **Current Date:** current date taken as a variable value.
- **Current Time:** current time taken as a variable value.
- **Counter:** variable that changes its value incrementally or decrementally with each label print.

TIP: All label or solution variables are managed in [Data Source Explorer](#).

Variable

Variable is a type of variable that obtains its value at print time.

General

About group of settings identifies the variable and sets its definition.

- **Name:** unique variable name. This name is used as the variable reference during its use.

NOTE Avoid using non-alphanumeric characters when defining the variable name.

Enter the name to make the variable easy to find when listed among other variables in the data source explorer.

- **Description:** is a field that allows adding additional information and suggestions.

Definition group of settings defines which input data types are valid for a variable.

- **Data type** defines what type of data is stored in a variable.
 - **Text:** variables that contain text.
 - **Date:** variables that contain date values.
 - **Time:** variables that contain time values.
 - **Floating point:** representation of real numbers in a variable.
 - **Currency:** variables that contain monetary values.
- **Initial value:** starting value that is assigned to a variable when created. It is defined using one of the following methods:
 - Manually entering a fixed value. Characters from any [group of allowed characters](#) are permitted.
 - Using a dynamic value. Dynamic data sources from the toolbar are supported – two options are available:

- Enter the source as **Name** in square brackets, e.g. [CurrentDate], [Counter].
- Select the dynamic data source from the drop down list.
- Using a [special character](#):
 - Special character can be entered manually using the less than/- greater than signs, e.g. <CR>, <LF> ...
 - Special character can be selected from the drop down [list](#).

NOTE Designer supports combined values as the initial value. Read more about combining the values [here](#).

EXAMPLE A combined initial value of a variable may contain a fixed value, a dynamic data source and special characters. The order of inserted items can be set randomly. Three options:

1. aaa123[Variable]<CR>
2. <CR>aaa123[Variable]
3. [Variable]<CR>aaa123

Make sure the inserted initial value meets the criteria defined with **Output Rules** for each data type.

Prompting group of settings defines the print time behavior of a data source. Read more about prompting [here](#).

Dynamic value group defines how the last used dynamic value of a variable is handled.

- **Remember the last used value (dynamic value):** Designer stores the last used value of a variable. The last used value is stored in an external text file at the same location as the label or solution file. Files that store the last used values have the same filename as the label or solution, followed by .DVB extension.

NOTE When sharing labels with dynamic values, make sure not to share only label or solution files (.NLBL or .NSLN), but also files that store last used dynamic values (.DDV).

NOTE Label or solution must be saved before enabling this option.

EXAMPLE The last used value is useful when the continuation of numbering from the last printed label is required (e.g. serial number). Counter's last value is stored and the numbering is continued from this point at next use.

Text

Text data type is used for variables that store textual content. As a result, only textual input is allowed as the variable input data type.

Input Rules

Data group defines permitted data properties.

- **Allowed characters:** definition of permitted variable input characters.

Groups of allowed characters for data input filtering are described in section [Groups of Allowable Characters](#).

- **Limit variable length:** maximum length of variable value.
- **Fixed length:** variable must contain the exact given number of characters as defined in the **Limit variable length**.

Output Rules

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties. **Output rules** set the final variable formatting – they define how the variable value is going to be presented in an object.

Date

Date data type stores date related values in the selected variable. Date field displays the date value using [various date formats](#). The date value format can be either selected from the preloaded formats, or customized to meet the specific local, regulatory or industry related requirements.

Input Rules

Input Formatting group defines the allowed date format and displays a preview.

- **Input format:** allowed date input format.
- **Sample value:** displays the preview according to the selected input format.

NOTE Designer supports a [range of preloaded or customized date formats](#).

Check range sets optional minimum and maximum date values.

- **Minimum value:** the earliest allowed date.
- **Maximum value** the latest allowed date.

Output Rules

Output formatting sets the output date format.

- **Output format:** format in which the date is displayed.
- **Output language:** language selection and regional formatting for days and months.

Output Language becomes relevant when the dates that include months or dates are written in words. In some cases, data calculations may be affected as well. For example, in US, a new week begins on Sunday whereas in EU and other countries, a new week begins on Monday.

- **Sample value:** date preview according to the selected input format. Current date is shown if the initial value is not defined.

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties.
Output rules set the final variable formatting – they define how the variable value is going to be presented in an object.

Time

Time data type stores time values in a variable. Time field displays the date value using [various time formats](#). The time value format can be either selected from the preloaded formats, or customized to meet the specific local, regulatory or industry related requirements.

Input Rules

Input Formatting defines the allowed time format and displays a preview.

- **Input format:** allowed time input format.
- **Sample value:** variable preview according to the selected input format.

NOTE Designer supports a [range of preloaded or customized time formats](#).

Check range defines the minimum and maximum time values. Defining the minimum and maximum limits is optional.

- **Minimum value:** the earliest allowed time value.
- **Maximum value:** the latest allowed time value.

Output Rules

Output formatting defines the output time format.

- **Output format:** format in which the time is displayed.
- **Sample value:** time preview according to the selected input format.

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties.
Output rules set the final variable formatting – they define how the variable value is going to be presented in an object.

Currency

Currency data type is used for variables that store numerical values of monetary amounts. Define currencies for various regions and set their properties.

Initial Value Definition

Initial value for **Currency Data type** is defined using one of the following methods:

- Manually entered fixed value. The number is delimited according to the **Input formatting** settings.
- Use of a dynamic value. Dynamic data sources from the toolbar are supported – two options are available:
 - the source is entered **Name** in square brackets, e.g. [Variable_1].
 - dynamic data source selection from the drop down list.
- Use of a special character:
 - Special character can be entered manually using the less than/greater than signs, e.g. <CR>, <LF> ...
 - Special character can be selected from the drop down [list](#).

NOTE Designer supports combined values as the initial value. Read more about combining the values [here](#).

EXAMPLE A combined initial value of a variable may contain a fixed value, a dynamic data source and special characters. The order of inserted items can be set randomly. Three options:

1. aaa123[Variable]<CR>
2. <CR>aaa123[Variable]
3. [Variable]<CR>aaa123

Input Rules

Input formatting group specifies the allowed input currency format.

Decimal delimiter is the character that separates the integer part from the fractional part of value written in decimal form.

Decimal places is the number of decimal places that is allowed to be included in the value.

Use 1000 separator enables using a separator that groups the thousands into groups.

- **Separator:** a character that is used as 1000 separator.

Currency symbol is a graphic symbol that represents a currency.

- **Placement:** position of the currency symbol.

Sample value displays a preview of the currency input format.

Limit variable length enables limiting the number of digits to be defined in a variable.

- **Length (characters):** allowed number of digits in a variable.

Check range defines the minimum and maximum values expressed in currency. Defining the minimum and maximum limits is optional.

- **Minimum value:** the lowest allowed input currency value.

NOTE If already defined, the initial value is taken as the minimum value.

- **Maximum value:** the highest allowed input currency value.

Output Rules

Input formatting specifies the preferred output currency format.

- **Decimal delimiter:** character that separates the integer part from the fractional part of a value written in decimal form.
- **Decimal places:** number of decimal places to be included in the value.
- **Use 1000 separator:** separator that groups the thousands into groups.
 - **Separator:** character that is used as 1000 separator.
 - **Sample value:** preview of the current number input format.
- **Currency symbol** is a graphic symbol that represents a currency.
- **Placement** defines the currency symbol's position. Select it from the drop down list.
- **Sample value** displays a preview of the currency input format.

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.

- **On right:** adds pad characters on the right side of the data value.
- **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties. **Output rules** set the final variable formatting – they define how the variable value is going to be presented in an object.

Floating Point

Floating Point data type specifies the representation settings for numeric values that are stored in a variable. This **Data type** is used to set the digit grouping points (separators) according to the regional specifics, and to place the decimal delimiters at the right places.

Input Rules

Input formatting specifies the allowed input number format.

- **Decimal delimiter:** specifies the character that separates the integer part from the fractional part of a number written in decimal form.
- **Decimal places:** the number of decimal places to be included in the number.
- **Use 1000 separator:** a separator that groups the thousands into groups.
 - **Separator:** a character that is used as thousands separator.
- **Sample value:** displays a preview of the current number input format.
- **Limit variable length:** enables limiting the number of digits to be defined for a variable.
 - **Length (characters):** allowed number digits in a variable.

Check range defines the minimum and maximum number values. Defining the minimum and maximum limits is optional:

- **Minimum value:** the lowest allowed input number.

NOTE If already defined, the initial value is taken as the minimum value.

- **Maximum value:** defines the highest allowed input number.

Output Rules

Input formatting specifies the preferred output number format.

- **Decimal delimiter:** the character that separates the integer part from the fractional part of a number written in decimal form.
- **Decimal places:** the number of decimal places to be included in the number.
 - **Auto:** decimal places are defined by local system settings.
- **Use 1000 separator:** enabled use of a separator that groups the thousands into groups.
 - **Separator:** a character that is used as thousands separator.
 - **Sample value** displays a preview of the current output format.

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties. **Output rules** set the final variable formatting – they define how the variable value is going to be presented in an object.

Variable Keyboard Input

Variable Keyboard Input is a type of variable that enables the content of a prompted field to be different for every print job. Its value is defined before each printing.

General

Definition group of settings defines which input data types are valid for a variable.

- **Data type** defines what type of data is stored in a variable.
 - **Text**: keyboard input that contains text.
 - **Date**: keyboard input that contains date values.
 - **Time**: keyboard input that contains time values.
 - **Floating point**: representation of real numbers in a variable.
 - **Currency**: variables that contain monetary values.
- **Initial value**: starting value that is assigned to a variable keyboard input when created. It is defined using one of the following methods:
 - Manually entering a fixed value. Characters from any [group of allowed characters](#) are permitted.
 - Using a dynamic value. Dynamic data sources from the toolbar are supported – two options are available:
 - Enter the source as **Name** in square brackets, e.g. [CurrentDate], [Counter].
 - Select the dynamic data source from the drop down list.
 - Using a [special character](#):
 - Special character can be entered manually using the less than/greater than signs, e.g. <CR>, <LF> ...
 - Special character can be selected from the drop down [list](#).

NOTE Designer supports combined values as the initial value. Read more about combining the values [here](#).

EXAMPLE A combined initial value of a variable may contain a fixed value, a dynamic data source and special characters. The order of inserted items can be set randomly. Three options:

1. aaa123[Variable]<CR>
2. <CR>aaa123[Variable]
3. [Variable]<CR>aaa123

Make sure the inserted initial value meets the criteria defined with **Output Rules** for each data type.

Prompting group of settings defines the print time behavior of a data source. Read more about prompting [here](#).

Dynamic value group defines how the last used dynamic value of a variable is handled.

- **Remember the last used value (dynamic value):** Designer stores the last used value of a variable. The last used value is stored in an external text file at the same location as the label or solution file. Files that store the last used values have the same filename as the label or solution, followed by .DVV extension.

NOTE When sharing labels with dynamic values, make sure not to share only label or solution files (.NLBL or .NSLN), but also files that store last used dynamic values (.DDV).

NOTE Label or solution must be saved before enabling this option.

EXAMPLE The last used value is useful when the continuation of numbering from the last printed label is required (e.g. serial number). Counter's last value is stored and the numbering is continued from this point at next use.

Current Date

Current Date is a type of variable that displays the current date value. The value is obtained from system or printer clock.

General

About group of settings identifies the variable and defines date output format and language.

- **Name:** unique variable name. This name is used as a variable reference during its use.
- **Description:** is a field that allows adding additional information and suggestions.
- **Output format:** format in which the date is displayed.
- **Output language:** language selection and regional formatting for days and months.

EXAMPLE Output Language becomes relevant when the dates that include months or dates are written in words. In some cases, data calculations may be affected as well. For example, in US, a new week begins on Sunday whereas in EU and other countries, a new week begins on Monday.

- **Output preview:** displays how the printed current date looks like. The range of used characters adapts to the selected **Output language** and printer.

Date offset group enables adding or subtracting a certain number of days, months or years from the current date. The offset date is displayed in the object instead of the present date.

- **Days:** date offset in days.
- **Months:** date offset in months.
- **Years:** date offset in years.

Printer Clock group defines how the printer clock should be used as the date value source.

- **Always use computer clock:** computer (system) clock set as the exclusive **Current Date** value source.
- **Always use printer clock:** printer clock set as the exclusive **Current Date** value source. An error is reported if the printer clock is unavailable.
- **Use printer clock if supported:** printer clock set as the preferred **Current Date** value source. If the printer clock is not supported, the computer (system) clock value is used instead.

NOTE The selected clock source option defines the range of allowed date **Formats**. Printer clock option only allows the use of printer supported date formats. An error is reported if a non-valid format is used. Computer (system) clock option allows using [a range of preloaded or customized date formats](#).

Output Rules

Prefix and Suffix values may be added to a variable value if required.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Multiline divides the text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties. **Output rules** set the final variable formatting – they define how the variable value is going to be presented in an object.

Date Formats

Designer enables flexible use of date fields. When defining the formats, the following notations are used:

Notation	Description
d	The number of day in a month. Occupies one or two characters.
dd	The number of day in a month. Always occupies two characters – leading zeros are added if necessary.
M	M is the number of month. Occupies one or two characters.
MM	MM is the number of month. Always occupies two characters.
yy or YYYY	The year represented with 2 or 4 digit numbers.
ddd	Abbreviation of the day of week name.
dddd	The full day of week name.

Notation	Description
MMMM	The full name of month.
MMM	The abbreviation of the name of month.
J	The number of days since 1. January. Occupies from one to three characters.
JJJ	The number of days since 1. January. Always occupies three characters.
W	The week number in current year. Occupies one or two characters.
WW	The week number in current year. Always occupies two characters.
N	The weekday number. The value range takes 1–7 characters, where 1 represents Monday and 7 represents Sunday.
custom text	Any sequence of characters is displayed unchanged. Insert dots, commas and other characters to present the date as required.

Date format examples

Format	Printed Date (English)
d.M.yyyy	10.3.2016
dd/MM/yy	10/03/16
dddd, d.MMMM yyyy	Thursday, 10.March 2016
JJJWWyyyy	069102005
textd/M/yyyytext	text10/3/2016text

Current Time

Current Time is a type of variable that displays the current time value. The value is obtained from system or printer clock.

General

About group of settings identifies the variable and defines time output format and language.

- **Name:** unique variable name. This name is used as a variable reference during its use.
- **Description:** is a field that allows adding additional information and suggestions.

Format group sets the current time presentation in an object.

- **Output format:** format in which the time is displayed.
- **Output preview** displays how the printed current time format looks like.

Time offset enables adding or subtracting a certain number of seconds, minutes or hours from the current time.

- **Seconds:** time offset in seconds.
- **Minutes:** time offset in minutes.
- **Hours:** time offset in hours.

Printer Clock group defines how the printer clock should be used as the time value source.

- **Use printer clock if supported:** printer clock set as the preferred current time value source. If the printer clock is not supported, the system clock value is used instead.
- **Always use printer clock:** printer clock set as the exclusive **Current Time** value source. An error is reported if the printer clock is unavailable.
- **Always use computer clock** computer (system) clock set as the exclusive **Current Time** value source.

NOTE The selected clock source option defines the range of supported time **Formats**. Printer clock option only allows the use of printer supported time formats. An error is reported if a non-valid format is used. Computer (system) clock option allows using [a range of preloaded and customized time formats](#).

Output Rules

Prefix and Suffix values may be added to a variable value if required.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Multiline divides the text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties. **Output rules** set the final variable formatting – they define how the variable value is going to be presented in an object.

Time Formats

Designer enables flexible use of time fields. Select a predefined time format or create a customized one. When defining the formats, the following notations are used.

h	Hours in 12-hour format. AM/PM is added if selected. Occupies one or two characters.
hh	Hours in 12-hour format. AM/PM is added if selected. Always occupies two characters. Leading zeros are added if necessary.
H	Hours in 24-hour format. Occupies one or two characters.
HH	Hours in 24-hour format. Always occupies two characters.
mm	Used for minutes.
ss	Used for seconds.

Time Format Examples

Format	Printed Date
h:mm {AM/PM}	8:25PM
H:mm	20:25
hh:mm:ss	08:25:36

Counter

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Counter is a type of variable whose value increments or decrements along with the changing value of system or printer counter.

Thermal printers are usually equipped with an internal incremental counter. This is a dedicated counter that counts the printed labels internally. The printer only receives the first value and automatically increases or decreases it on the subsequent labels. This option reduces the amount of data transferred between computer and printer as only initial value is sent to the printer. Internal counter speeds up the label production significantly.

General Tab

About group of settings identifies the variable and defines serialization details.

- **Name:** unique variable name. This name is used as a variable reference during its use.
- **Description:** is a field that allows adding additional information and suggestions.

Serialization group of settings defines the counter behavior.

- **Counter type:** counter value increasing or decreasing:
 - **Incremental:** value increases along with the printed labels.
 - **Decremental:** variable value decreases along with the printed labels.
- **Step:** amount of units that represent the next state of counter value.
- **Repetition:** number of repetitions for each counter value.
- **Initial value:** value that is used when the counter starts.
- **Preview:** displays the counter value sequence as defined by the current **Step**, **Repetition** and **Initial value**.

EXAMPLE Counter Step = 3, Repetition = 3 and Initial value = 1 are: 1,1,1,4,4,4,7,7,7,10, 10, 10, 13, 13, 13.

Prompting group of settings defines the print time behavior of a data source. Read more about prompting [here](#).

Dynamic value group defines how the last used dynamic value of a variable is handled.

- **Remember the last used value (dynamic value):** Designer stores the last used value of a variable. The last used value is stored in an external text file at the same

location as the label or solution file. Files that store the last used values have the same filename as the label or solution, followed by .DVV extension.

NOTE When sharing labels with dynamic values, make sure not to share only label or solution files (.NLBL or .NSLN), but also files that store last used dynamic values (.DDV).

NOTE Label or solution must be saved before enabling this option.

EXAMPLE The last used value is useful when the continuation of numbering from the last printed label is required (e.g. serial number). Counter's last value is stored and the numbering is continued from this point at next use.

Printer Counter defines how the printer counter should be used.

- **Use printer counter if supported:** printer counter is set as the counter of choice if supported by the active printer. If the printer counter is not supported, system counter is used instead.
- **Always use printer counter:** printer counter set as the exclusive counter value source. If the printer counter value is not available, the default (system counter) value is used.
- **Always use computer counter:** computer counter set as the only counter value source.

TIP: Input rules help the user when inserting the variable data. They act as a filter that defines the type, length and other input data properties.
Output rules set the final variable formatting – they define how the variable value is going to be presented in an object.

Thermal printers are in most cases equipped with an internal incremental counter. Such counter works as a dedicated device that counts the printed labels internally. The printer only receives the first value of the counter and then automatically increments the counter in steps of 1 on the subsequent labels.

TIP: This ability reduces the amount of data that needs to be transferred between computer and printer since only the starting value is sent to printer. This speeds up the label production significantly.

To use the counter as internal printer element follow the below listed rules:

The variable's maximum length is limited by the printer. The value should be included in the printer user guide.

TIP: If the exact maximum variable length value is not available, NiceLabel recommends making a few test prints for determining the value.

- Set variable length to fixed.
- Set variable format to numeric.
- Text object that is linked to the variable must be formatted using an internal printer font.
- Enable **Always use printer counter** option.

- Make sure the Internal Element icon is visible next to the counter text box.
- Make sure an internal printer font is used for the counter text box.

Input Rules

Data defines the counter input criteria.

- **Allowed characters:** permitted characters for variable values. Groups of allowed characters for data input filtering are described in section [Groups of Allowed Characters](#).

EXAMPLE Non-numeric characters can also be used as counter values. **Alphanumeric** sets the sequence with Step = 3 and Initial value = 1 as 1, 4, 7, A, D, G, J, M, P, S, V, Y, b, e, h, ...

- **Limit variable length:** maximum length of a variable value.
 - **Length (characters):** specifies the exact permitted number of characters.
- **Fixed length:** variable must contain the exact given number of characters as defined in the **Limit variable length**.

Check range group defines minimum and maximum counter values.

- **Minimum value:** minimum counter value.
- **Maximum value:** maximum counter value.

Rollover settings group defines the condition at which the counter automatically resets its value to default.

- **Using min/max:** minimum and maximum counter values activate the rollover.
- **When the selected data source changes:** data source value change activate the rollover.
- **When date or time changes:** date or time value change activate the rollover.

Output Rules

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

EXAMPLE Pad character is in most cases zero (0) added on the left side of the variable value. If the maximum variable length is set to 5 characters and the value is 23, the padded result is 00023.

Multiline: divides text into multiple lines.

WARNING Avoid using this setting if possible. The recommended alternative for presenting multiline text on a label or form is to use the [Text Box object](#).

- **Number of lines:** maximum number of lines for a variable value.
- **Line length:** maximum number of characters in a single line.
- **Word wrap:** divides the text into multiple lines at space character locations.

Variable Prompting

When designing labels with variables, a value has to be assigned to them before printing. Prompted variables have their values manually assigned at print time. The user is asked for the value of every variable before each print job.

The values are entered manually. The order in which they are entered may be specified using the [Prompt order](#) dialog.

Prompting group asks the user for manual data input – this is done after the print dialog opens.

- **Prompt at print time:** enabled or disabled prompting.

NOTE If a dynamic data source is included in the **Initial value**, prompting becomes disabled.

- **Prompt text:** text that is displayed to the user. This text serves as an instruction on what kind of values should be inserted before printing.
- **Value required:** variable value status – mandatory or optional. If the prompt text is left empty in case the value is set as mandatory, printing cannot start. An error message appears.

Functions As Dynamic Data Source

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

The purpose of functions is to manipulate the data that is assigned to label objects. Functions process the existing data source values and store the result in function-generated data sources.

Each function can be directly connected to an object and used as a part of another function.

TIP: All label or solution functions are listed in the [Data Source Explorer](#). Before defining the first function, the data source explorer contains no items.

Click **Add new function** and select the appropriate function from the list.

Designer includes the following function types:

- [Subset](#): extracts a specific part of data according to the user-specified rules.
- [Concatenate](#): merges two or more data source values into a single value.

- [Date Offset](#): offsets the present date.
- [Linear](#): transforms the current value using multiple types of linear functions.
- [VBScript](#): allows performing complex value transformations.
- [VBScript Expression](#): is a simplified version of VBScript.
- [Python Script](#): allows performing complex value transformations.
- [HIBC](#): encodes the data in compliance with the health industry barcode standard.
- [GS1-128](#): encodes the data in compliance with the GS-128 barcode standard.
- [ANSI MH10.8.2](#): encodes the data in compliance with ANSI MH10.8.2-2006 standard.
- [Transfer Data Syntax for High Capacity ADC Media](#): enables the ADC users to use a single mapping utility, regardless of which high-capacity ADC media is employed.
- [Read from file](#): function reads content from a specified file and displays it in an object.
- [NDEF Message](#): function allows you to define a message encapsulation format for the exchange of data information over an Near Field Communication (NFC) link.

Subset

Subset function extracts a specific part of data according to the user-specified rules.

About group identifies the function.

- **Name**: function ID, initially defined by the function type.
- **Description**: function's purpose and role as defined by the user.

Input data source defines the existing or newly added input data source (variable, function or database record). The final (output) value is extracted from the selected input data source value.

Definition group offers two methods for extracting the data from input data source.

Fixed length extracts a fixed number of characters from the input data source.

- **Offset**: number of characters to be skipped from the beginning of the value.
- **Length**: length of extracted value.

EXAMPLE

Input value: ABCDE
 Offset: 0
 Length: 3
Subset value: ABC

EXAMPLE

Input value: ABCDE
 Offset: 2
 Length: 3
Subset value: CDE

Delimiter is used when extracting the data, separated by the user-specified delimiter.

- **Delimiter:** character (comma by default) that separates input value fields.
- **Text qualifier:** character that encloses the values within the fields (quotation mark by default).

If a text qualifier is used, the delimiter within the text qualifiers also belongs to the data value. Text qualifier can be left empty.

- **Field number:** the field number that is extracted from the input data source.

EXAMPLE

Input value: "A", "B", "C", "D"
 Delimiter: ,
 Text qualifiers: "
 Field number: 3
Subset value: C

Concatenate

Concatenate function merges two or more data source values into a single value.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Input data source group defines the existing or newly added input data source (variable, function or database record) or fixed text that will be used in the function.

Output Options group defines the output value format.

Delimiter is a character that is inserted between the concatenated values. The delimiting character can be entered manually or selected from one of the additional options:

- **New Line (CR/LF):** new line character.
- **Insert special character:** [special character](#) is entered.

NOTE Delimiter is an optional value. With no delimiter defined, the concatenated values are merged without a delimiting space or character.

- **Ignore empty values:** ignores empty data source values. These values are excluded from concatenation.

This option is useful if you want to avoid duplicated delimiters if empty values appear.

EXAMPLE

Data source value 1: A, B, C, D
 Data source value 2: <empty>
 Data source value 3: E, F, G
 Delimiter: ,
Concatenated value with vs. without Ignore empty values: A, B, C, D, E, F, G vs. A, B, C, D,, E, F, G

NOTE **Ignore empty values** option is effective only after executing a print command. When storing a label in [store/recall printing mode](#) or when exporting a label, the empty values are not ignored. Delimiters appear duplicated.

Date Offset

Date Offset function defines the number of days, months and/or years to be added to or subtracted from any specified date (might be the current date or any past/future date).

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Input data source defines the input data source from which the data will be used in the function.

Offset is the number of days, months or years to be added to or subtracted from the input data source.

EXAMPLE

Current date: March 8 2016

Offset: Days +1; Months +1; Years +1

Result: April 7 2017

Output formatting group defines the function's output.

- **Output format:** defines the date format to be used in the connected object.
- **Sample:** current date in the selected **Output format**.

Linear

Linear function multiplies or divides an input data source value by a defined **Multiplier**. An optional value (**Addition**) can also be added.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Input data source defines the input data source from which the data will be used in the function.

Linear function parameters are:

- **Mode:** linear function type.
- **a:** function multiplier.
- **b:** function addition.

Output formatting group defines the function output format.

Output format is the format of a modified value. **more...** opens additional formatting options:

- **Decimal separator:** character that marks the border between integral and fractional parts of a decimal numeral.
- **Decimal places:** number of places behind the decimal separator.
- **Use 1000 separator:** thousands separated or non-separated by a delimiter.
- **Delimiter:** character that separates the thousands from the rest of the numeral.
- **Sample:** preview of the formatted output value.

EXAMPLE

Input value: 123

a: 2

b: 20

Output value: $2 * 123 + 20 = 266$

VBScript

VBScript function enables reading, writing, and manipulating the data that belongs to any of the connected data sources.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

VBScript group allows defining the script.

- **Verify:** validates the entered script syntax.
- **Export:** exports the script to an external file that can be imported to other applications.
- **Import:** imports a VBScript from an external file.

NOTE The result of the script must be saved in the 'Result'. The value of 'Result' is inserted into the name of the function. Such function is available as a dynamic data source for further use.

EXAMPLE

The variable **NAME** provides the first and the last name of a person. Visual Basic Script function should break the names apart and use only the first name as the result of the function.

NAME variable initial value: **John Doe**

```
Dim Spc
Spc = InStr(NAME, " ")
if NAME <> "" then
    Result = Mid(NAME, 1, Spc-1)
end if
```

Result of VBScript: **John**

VBScript Expression

VBScript Expression is a simplified online version of [VBScript](#). This Designer function can be used to:

- manipulate existing variables
- extract sub-strings
- perform quick calculations

VBScript Expression reduces the need to write dedicated VBScripts. Instead of writing an entire script, insert a single-line expression in the edit field that is validated at print time.

TIP: When compared to VBScript commands, a VBScript expression command does not require the final value to be stored in **Result**.

About

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Visual Basic Expression

- **Verify:** validation of the entered script syntax.
- **Editor:** field for script writing and editing.

Python Script

Python script function supports even the most complex data manipulations on a label or a form.

TIP: When compared with VBScript, it proves out to be a more suitable option for 64-bit systems. It is also proved to be a notably faster scripting alternative.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Python Script

- **Verify:** validation of script syntax.
- **Editor:** dialog for writing and editing the scripts. **Editor** opens a new window that allows creating a Python Script.
- **Export:** allows exporting and reusing the script in other applications.
- **Import:** imports scripts from external applications to be used in NiceLabel Designer.

EXAMPLE:

The variable **NAME** provides the first and the last name of a person. Python Script function should break the names apart and use only the first name as the result of the function.

NAME variable initial value: **John Doe**

```
name = NAME.Value
SpC = name.find(' ')

if name != '' and SpC != -1:
    Result.Value = name[0:SpC]
else:
    Result.Value = name
```

Result of Python script: **John**

HIBC

HIBC is a barcode standard used specifically in health industry, as directed by the HIBCC organization. This standard supports composite bar codes and supports the use of multiple items such as item codes, quantity and batch number in a single barcode.

TIP: Visit [HIBCC](#) website for more information about the standard.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Structure group selects one of the three available HIBC barcode **Types** to be used:

- **Primary:** mandatory data structure with a fixed structure which identifies supplier and item.
- **Secondary:** optional data structure which is indicated using the separator sign "/". It may have a variable (but prescribed) structure to contain serial or batch numbers, quantity and expiration date.
 - **Primary definition:** necessary item when defining the **Secondary** data structure. The three **Primary** data fields of a HIBC function must be added to the **Secondary** data structure. **Primary definition** selects the appropriate existing HIBC function.
- **Concatenated:** merges the first two structure types into a single data structure.

Definition group defines the content of HIBC barcode fields:

Primary data structure fields:

- **Labeler ID code (LIC):** field assigned and maintained by the HIBCC. The first character of this field is always an alphabetic character. The LIC may identify a labeler to the point of separate subsidiaries and divisions within a parent organization.
- **Product or Catalog...:** compressed product or catalog number.
- **Unit of Measure...:** numeric representation of packaging level (0 to 9) with 0 being the lowest level or "unit-of-use".

EXAMPLE A company might pack unit-of-use items in a box, boxes in a carton, and cartons in a case. One way of labeling would be, unit-of-use = 0; Box = 1; Carton = 3; and Case = 5.

Secondary data structure fields:

- **Quantity:** two- or five-digit field describing the number of units-of-use included in the package identified by the bar code label.
- **Date format:** preferred date format to be used with a HIBC label. If no date should be included on a label, select one of the formats that contain "No date".
- **Date:** displays the present date.
- **Lot/Batch:** field can be alphanumeric and may vary in length to up to a maximum of 18 characters. If the field is not required, it should be left empty.
- **Serial number:** field can be alphanumeric and may vary in length to up to a maximum of 18 characters. If the field is not required, it should be left empty.
- **Production date: Data Identifier** formatted as YYYYMMDD.

GS1-128

GS1-128 function encodes barcode data using the GS1-128 standard. The standard supports encoding of textual data, numbers, functions, and the entire set of 128 ASCII characters.

GS1-128 encodes the data and defines its meaning by defining a list of **Application Identifiers** (AI). These identifiers define the content and length of the data they include.

Als include a data field that contains a fixed or variable number of characters.

TIP: For more information about GS-128 standard and encoding principles, visit the GS1 [website](#).

The list of available Als is available [here](#).

About

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Application Identifiers field displays the selected AIs.

Edit Function Definition button opens a dialog for editing the identifiers. **Function Definition** dialog allows the user to **Add, Delete, Move**, and edit the selected identifiers. There are four columns with identifier properties:

- **Identifier:** column with identifier AI number and description.
- **Value:** column with a manually or dynamically defined value as given by the selected **Data source**.

Each **Value** column allows a limited number of characters to be inserted. The limitation is defined by the standard and varies according to the selected identifier.

- **Options:** column with additional identifier options (if available).

Delimiter group defines the delimiting characters for separating the AIs.

A single barcode may include multiple AIs. These fields are separated using left and right **Delimiter**. By default, first two digits of AI are used. Custom delimiters may be defined by inserting alphanumeric characters.

Additional function outputs group defines a subordinate function.

- **Create output function with unformatted contents** creates a subordinate function that uses the unformatted data encoded by the parent GS1-128 function.
- **Function name:** the name of the newly created subordinate function.

ANSI MH10.8.2 (ASC)

ANSI MH10.8.2 (ASC) function encodes barcode data using the ANSI MH10.8.2-2006 standard. This standard provides a range of MH 10/SC 8 data identifiers and GS1 application identifiers. It enables the assignment of new data identifiers, and defines the correlation, or mapping of data identifiers to application identifiers.

TIP: For more information about ANSI MH10.8.2 (ASC) standard, visit the [official web-site](#).

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Application Identifiers enable cross-industry standardized use of data identifiers. They are used with any alphanumeric data carrier.

Edit Function Definition button opens the **Function Definition** dialog. It allows the user to **Add, Delete, Move**, and edit the selected identifiers.

There are three columns with identifier properties:

- **Identifier:** column with identifier ID.
- **Value:** column with manually inserted value or an automatically defined value as given by the selected **Data source**.

NOTE Each **Value** column allows a limited number of characters to be entered. The limitation (format) is defined by the standard and varies according to the selected identifier.

Transfer Data Syntax For High Capacity ADC Media

High-capacity automatic data capture (**ADC**) technologies, such as two-dimensional symbols, RFID transponders, contact memories, and smart cards, encode multiple fields of data. These fields are usually parsed by the information system and mapped to the specified data fields.

ISO/IEC 15434:2005 defines the syntax for high-capacity ADC media. This enables the ADC users to use a single mapping utility, regardless of which high-capacity ADC media is employed.

The data encoded according to ISO/IEC 15434:2005 includes:

- data for shipping, receiving, and inventory of transport units
- data for supporting documentation related to unit loads or transport packages
- data for sorting and tracking of transport units

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.
- **Application Identifiers:** cross-industry standardized set of data identifiers.

TIP: The purpose of identifiers is to provide a unique item identification. To manage the identifiers, click **Edit Function Definition**.

Function Definition dialog allows the user to **Add**, **Delete**, and edit the selected identifiers.

- **Format Envelope:** column defines the starting and ending positions for a data item in a given **Format**.
- **Data Elements:** column defines the identifier content by inserting the data to be encoded.

NOTE Each identifier allows adding multiple elements.

- **Format Header Data:** defines two mandatory format header elements.
 - **Version:** organization that controls the data structure.
 - **Release:** release number of ADC media standard.

EXAMPLE :

- 02 is compliant with ASC MH10/SC 8 (using measurement qualifiers of pounds and kilograms)
- 06 is compliant with International Air Transport Association (IATA) rules.
- 56 is compliant with International Federation of Freight Forwarders (FIATA) rules.
- 96 is compliant with ASC MH10/SC 8 (pounds only).

Read From File

Read from file function reads content from a specified file. The file can be accessible locally or remotely via network connection.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

Read from file parameters group sets the file connection details.

File name sets the file connection.

Encoding specifies the encoding type for the sent data.

- **Auto:** automatically defined encoding .

If needed, select the [preferred encoding type](#) from the drop down list.

NDEF Message

NDEF Message function allows you to define a message encapsulation format for the exchange of data information over an Near Field Communication (NFC) link. Such link is established between two NFC devices, or between an NFC device and a tag.

TIP: NFC is a set of communication protocols that enable two devices to establish communication by bringing them within 4 cm (2 in) of each other.

NDEF message encapsulates one or more application-defined records which appear in a variety of types and sizes. These records are combined into a single message.

About group identifies the function.

- **Name:** function ID, initially defined by the function type.
- **Description:** function's purpose and role as defined by the user.

NDEF Message Structure group displays the NDEF records that are included in the message.

Click **Edit Function definition** to open the **NDEF Message dialog**. This dialog allows the user to **Add, Delete, Move**, and edit the NDEF records. There are two columns with record properties:

- **NDEF Record Type:** identifies the record type. The listed standard record types are available in Designer:
 - **Uri:** contains a string of characters that identifies a web resource.
 - **Text:** contains textual content with information about text encoding and language code.
 - **Smart Poster:** includes multiple sub-records – URI, title, recommended actions, icon, size and type.

NOTE Smart Poster content is represented as a single record content, although internally the structure is created as multiple (sub)records within a single record.

- **BlueTooth Handover Select:** a set of records including various items – handover version, device address string, complete local string, class of device, and service class.
- **Custom:** record type which allows encoding the non-native NFC data.

Drag and drop the records in NDEF Message dialog to quickly change their position.

Detailed descriptions of NDEF record types are available in [NFC Forum technical specifications](#).

- **Record Definition:** settings as defined by the NDEF standard. Available options depend on the selected record type.

Include capability container adds capability container to the encoded data. Capability container stores control data for managing the NFC data in a tag or a device. It tells the NFC device that the received data is an NFC message. In cases when NFC content needs to be encoded into a standard high frequency (HF) RFID tag, enable the **Include capability container** option. This signals the reading device that NFC content is stored in the tag. Certain NFC compliant tags already include capability container in the tag which means that there is no need for including it as a part of the generated content.

Databases As Dynamic Data Source

PRODUCT LEVEL INFO Creation of forms and use of form objects is available in NiceLabel PowerForms.

Databases can be used as dynamic data source for label or form objects. To make the database content accessible and retrievable from the selected object, the database connection must be properly established and configured.

The most time efficient and user friendly way of adding a database to your label or solution data sources is to use the [Step-by-Step Database Wizard](#).

Designer also allows the database connections to be established and configured manually. This way, the entire range of connection settings becomes configurable. It is recommended that only experienced users choose this option.

All label or solution databases are listed in the [Data Source Explorer](#).

Designer supports a wide selection of database types. The supported database types are listed [here](#).

Read about how to connect to the supported database types [here](#).

Read about other available object data sources and how to use the Dynamic Data Manager [here](#).

Supported Database Types

Designer supports multiple types of databases:

- Microsoft Excel
- Microsoft Access
- Microsoft SQL Server
- Text File databases
- Oracle databases

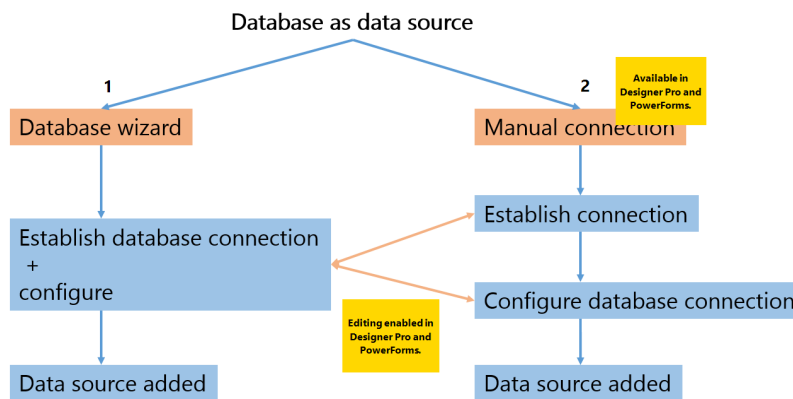
- MySQL
- OLE databases
- ODBC data source

TIP: NiceLabel recommends using standard database types such as Text, Excel, Access, SQL Server, and MySQL. The use of standard database types is easier and more time efficient due to optimized application performance and user interface. When working with a non-standard database type, use the OLE and ODBC options.

Read about how to connect to the supported database types [here](#).

Database Connection Options

Designer offers two ways for connecting an object to a database. The diagram below shows the two available options.



1. Step-by-step Database Wizard offers a guided process for:

- connecting a database to a label or form object
- adding a database to the labeling solution's data sources

The process of establishing and configuring a wizard based database connection is described [here](#).

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

2. Manually established and configured database connection.

NOTE This option is intended for advanced users. It allows detailed configuration and offers all of the available database connection settings.

The process of establishing and configuring a database connection manually is described [here](#).

Step-by-Step Database Wizard

[Database wizard](#) is a guided process that allows the user to configure a connection to a database and to select which tables and fields will be used. Dedicated buttons provide instant access to the most commonly used database types. Use the **All Databases** button to start the wizard in general mode and to select the database type during the next step.

[Edit Database](#) allows you to edit all existing connected databases using a wizard.

The wizard additionally allows you to sort, filter records, and to define how many label copies will be printed per database record.

Adding A Database

There three options for starting the **Database Wizard**:

- Option 1: Click the preferred database button in **Designer Data tab ribbon -> Step-by-step Database Wizard** group.
- Option 2: Click the preferred database button in **Dynamic Data Manager -> Step-by-Step Database wizard** ribbon group.
- Option 3: Click the **+Database wizard** command in [Data Source Explorer](#) or object properties.

Below listed are the available wizard options. To successfully add a database, follow the steps for each database type:

- [Adding an Excel database](#)
- [Adding an Access database](#)
- [Adding an SQL Server database](#)
- [Adding a Text File database](#)
- [Adding an Oracle database](#)
- [Adding a MySQL database](#)
- [Adding other OLE database](#)
- [Adding an ODBC data source](#)

Database Wizard for Excel Database

This section describes how to add an Excel database to a form or label object using the Designer Step-by-Step Database Wizard.

Step 1: Connection Settings

This step defines the database connection parameters.

File name defines the database file location.

Advanced Setup opens the system configuration dialog. *Data Link Properties* window allows you to set the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **Next**.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as label or form object data source.

Database Wizard for Access Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to add an Access database to a form or label object using the Designer Step-by-Step Database Wizard.

Step 1: Connection Settings

This step defines the Access database file connection details.

File name selects the database file.

Authentication requires **User name** and **Password** for password protected Access database files.

Advanced Setup opens the system configuration dialog. *Data Link Properties* window allows you to set the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **Next**.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as label or form object data source.

Database Wizard for Microsoft SQL Server Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to add a Microsoft SQL Server database to a form or label object using the Designer Step-by-Step Database Wizard.

Step 1: Connection Settings

This step defines the database file connection details.

Server defines the database server.

Authentication group defines user authentication type for database server.

- **Use Windows authentication.** This option defines the Windows authentication as verification method for connecting to an SQL server. The user connects to a database using domain username and password.
- **Use SQL Server authentication.** This option defines the database user name and password as the verification method. To establish a connection, enter the user name and password provided by the database administrator.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard inputs in the dialog box.

Database Selection group selects the database on the connected server.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **Next** to proceed.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as label or form object data source.

Database Wizard for Text File

This section describes how to use a text file as data source in label or form objects. A text file is connected to an object using the Designer Step-by-Step Database Wizard.

Step 0: Text File Structure Wizard

Text File Structure Wizard window opens if a structure for a text file you are connecting hasn't been defined previously.

The steps for completing the **Text File Structure Wizard** are described in a [dedicated section](#).

NOTE After finishing this procedure, a text definition .sch file with the same name as the text database file and is created in the same folder. Next time the wizard is used on the same file, this procedure is no longer required.

Step 1: Connection Settings

This step defines the text file path.

File name defines the location of the Text file to be used. Enter the location manually or click **Browse** to locate it in the system.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **Next**.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

NOTE Table selection is not available when adding a text file as a database. The entire text file is treated as a single database table.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Fields tab displays available and selected database fields. Step 3 settings of this section can be redone on this tab.

Data Retrieving tab defines how the data should be retrieved from the connected database file. Read more about data retrieving [here](#).

Click **Finish**. The database is ready to be used as label or form object data source.

Database Wizard for Oracle Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to add an Oracle database to a form or label object using the Designer Step-by-Step Database Wizard.

Step 1: Connection Settings

This step defines the database connection details.

NOTE Oracle Database Provider is required to establishing a connection with Oracle database.

Data Source defines the Oracle Data Source name.

Authentication provides user name and password for establishing the connection.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard inputs or **Advanced Setup** dialog.

Advanced Setup button opens the *Data Link Properties* window allowing the user to define the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as a label or form object data source.

Database Wizard for MySQL Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to add a MySQL database to a form or label object using the Designer Step-by-Step Database Wizard.

Step 1: Connection Settings

This step defines the MySQL database connection details.

Database defines the exact database on a server.

Host defines the database address.

Port defines the port of the database server.

Authentication provides user name and password for establishing the connection.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as a label or form object data source.

Database Wizard for Adding Databases via OLE DB

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to add various types of databases via OLE DB source to a form or label object using the Designer Step-by-Step Database Wizard.

The OLE DB extracts data from a variety of OLE DB-compliant relational databases by using a database table, a view, or an SQL command.

EXAMPLE OLE DB can extract data from tables in Microsoft Access or SQL Server databases.

Step 1: Connection Settings

This step defines the OLE DB connection details.

Provider defines the provider to be used for accessing the data by exposing the OLE DB interfaces.

Authentication provides user name and password for establishing the connection.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Authentication provides user name and password that are used for the connection.

Advanced Configuration options are:

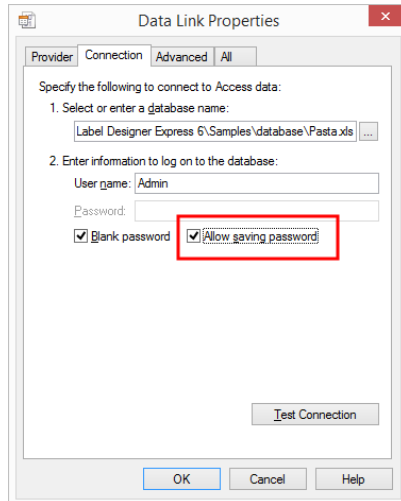
- **Automatically browse for tables** automatically displays the available OLE database tables. If this checkbox is cleared, you will have to enter the table name manually.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard inputs or **Advanced Setup** dialog.

Advanced Setup button opens the *Data Link Properties* window allowing the user to define the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

NOTE When connecting to a password protected database, make sure the **Allow saving password** option is selected. If not, even after a successful **Test Connection** procedure, database access is not going to be granted.



Test Connection button starts a connection testing procedure to confirm if a connection with database has been established successfully. A confirmation or error message appears.

Click **Next**.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as a label or form object data source.

Database Wizard for ODBC Data Sources

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

This section describes how to work with DesignerStep-by-Step Database Wizard when adding an ODBC data source.

The Microsoft® ODBC Data Source Administrator manages database drivers and data sources. This application is located in the Windows Control Panel under Administrative Tools.

For information about detailed ODBC Administrator procedures, open the [ODBC Data Source Administrator](#) dialog box and click Help.

Step 1: Connection information

This step defines the database connection details.

Data Source defines the source to retrieve the data from.

Driver displays the database driver according to the selected data source.

ODBC Administrator button opens the system ODBC administration dialog. See more details about the dialog [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Step 2: Tables and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 3: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 4: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 5: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**. The database is ready to be used as a label or form object data source.

Editing A Database

Edit Database button starts the Step-by-Step Database Wizard for configuring an existing database.

To properly reconfigure a database that has already been added, follow the below listed steps.

Step 1: Define Database Table

Use this step select among the existing databases. Select the database and the table you wish to edit. Click **Next** to proceed.

Step 2: Connection Settings

This step defines the database connection parameters.

File name defines the database file location.

Advanced Setup opens the system configuration dialog. *Data Link Properties* window allows you to set the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **Next**.

Step 3: Table and Fields

This step defines which database table and which fields of this table should be used as data source.

Available fields lists the available fields (columns) in the selected database.

Selected fields lists the fields (columns) which will be included in the database table.

Click **Add >** or **< Remove** buttons to add or remove the fields from the **Selected fields**.

NOTE When editing an existing database, a field cannot be removed if used in a script, function, action, or connected to a label or form object.

Click **Next**.

Step 4: Label Copies Per Record

This step specifies the number of label copies to be printed for each database record.

Fixed number of printed labels lets you insert the number of copies manually.

Dynamically defined number of printed labels sets the number dynamically using a data source value.

EXAMPLE The number of printed labels is defined in the database field of the record that is going to be printed.

EXAMPLE The number of printed records can be defined using a variable value. Its value may be set in another label or form object.

Click **Next** to proceed or **Finish** to continue working with the object.

Read more about how to define the number of printed copies [here](#).

Step 5: Create Objects

This step decides whether new objects that display the content retrieved from database fields should be added to a label/form or not.

Create Objects step is visible when:

- starting the database wizard from Designer **Data** tab ribbon and adding a new database by clicking the database button
- starting the wizard in [Data Source Explorer](#) or using a generic object **Add database** selector

TIP: The **Create Objects** step differs if you are adding a database while designing a label or a form. See the differences below.

Create Objects step for label designing:

- **Create a label text object for each field:** adds a [Text](#) object that contains database field content.
- **Do not create any label objects:** skips adding new objects.

Create Objects step for form designing:

- **Create an edit field object for each field:** adds an [edit field](#) object to the form. The added object(s) contains database field content.
- **Create a form table object:** adds a [database table](#) object to a form. The added object(s) contains database field content.
- **Do not create any label objects:** skips adding new objects.

NOTE The number of added objects depends on the number of fields in the database.

Click **Next**.

Step 6: Data Preview and Other Table Settings

This step gives a preview of the data retrieved from the database. It also offers additional table settings such as filtering and sorting.

Data tab displays a preview of data retrieved from the database file. You can use search controls at the top of the preview section to find a specific record.

NOTE Data preview shows up to 1000 rows.

Filter tab filters out the database file records. It allows you to define filtering conditions to be used when retrieving the data.

- **Add condition:** specifies single line condition(s) that filters out the content that meets the set criteria.
- **Add group:** specifies group(s) of conditions that filter out the content that meets the set criteria.

Sorting tab allows you to sort the retrieved data. Sorting is done for all of the fields that are added to the sorting list. Each field can be in ascending or descending order.

Data Retrieving tab defines how the data should be retrieved from the connected database file.

SQL tab offers a preview of the generated SQL statements.

Read more details about database table configuration [here](#).

Click **Finish**.

Manual Database Connection Setup

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

To establish a database connection manually gives you complete control over database connection settings and configuration options.

NOTE This option is intended for advanced users. It allows detailed configuration and offers all of the available database connection settings. NiceLabel recommends using the Database Wizard.

Manual database connections are done in three steps. First step is to create the connection. Second step is to choose which database tables will be used. Third step would be to configure the tables.

To connect to a database manually, follow the procedures described in the below listed topics:

- [Connect to Text File](#)
- [Connect to Microsoft Excel File](#)
- [Connect to Microsoft Access File](#)
- [Connect to Microsoft SQL Server File](#)
- [Connect to Oracle Database](#)
- [Connect to MySQL Database](#)
- [Connect to OLE Database](#)
- [Connect to ODBC Data Source](#)

Connect To Text File

Text File database can be used as a dynamic data source for [label objects](#) or [form objects](#).

Text files require some additional work before they are transformed into a "real" database. At start, any text file contains data values but has no information about the data structure, name fields, and maximum field lengths. These missing parameters need to be specified before the text file turns into a database which can be used as an object data source.

EXAMPLE

A widely used text database example are .csv files. In a .csv file, a delimiter separates the database fields. Each line provides the data for a single label – therefore, it can be understood as a "record" in database nomenclature.

To manually connect an object to a text file database, complete the following steps.

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Text File** as the preferred database type.

Click **OK**.

Step 2: Set Connection Information

Connection Information window defines the database file details.

File name defines the file location.

Test Connection button starts a connection testing procedure. It shows whether or not a connection with the database has been established. A confirmation or error message appears depending on the connection status.

Step 2a: Finish Text File Structure Wizard

Text File Structure Wizard window opens if a structure for a text file you are connecting hasn't been defined previously.

The steps for completing the **Text File Structure Wizard** are described in a [dedicated section](#).

NOTE After finishing this procedure, a text definition .sch file with the same name as the text database file and is created in the same folder. Next time the wizard is used on the same file, this procedure is no longer required.

Step 3: Define Database Connection Settings

Read how to set database connection details for text files [here](#).

Step 4: Define Database field Settings

Read how to set the connected database field properties [here](#).

Connect To Microsoft Excel File

Microsoft Excel databases can be used as a dynamic data source for [label objects](#) or [form objects](#).

To manually connect an object to an Excel file database, complete the below listed steps.

Step 1: Create a database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Microsoft Excel** as the preferred database type.

Connection Information window defines the database file details.

File name defines the file location.

Advanced Setup button opens the **Data Link Properties** window. This window allows you to define the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It shows if a connection with the database has been established successfully. A confirmation or error message appears depending on the connection status.

Click **Next**.

Step 2: Setting the connection name and description

Connection name defines the name for the connected database file. By default, it displays the filename of the connected file. Insert a new name to make it easy to be found in the Designer **Data Source** explorer.

Description is a field that allows adding additional information and suggestions for the connected database.

Connection identifies the currently connected database file. To replace the currently connected file, click the **Connection Setup** button. **New Database Connection Properties** window reappears – repeat step 1 to connect to an alternative database file.

Step 2: Add or remove Database Tables

Read how to set database connection details for Excel files [here](#).

Step 3: Set Table Configuration Details

Read how to configure the connected table [here](#).

Connect To A Microsoft Access File

Microsoft Access databases can be used as a dynamic data source for [label objects](#) or [form objects](#).

To manually connect an object to an Access file database, complete the following steps.

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Microsoft Access** as the preferred database type.

Click **OK**.

Step 2: Set Connection Information

Connection Information window defines the database file details.

File name defines the file location.

Authentication provides **User name** and **Password** for connecting to a protected file.

Advanced Setup button opens the **Data Link Properties** window. This window allows you to define the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).

Test Connection button starts a connection testing procedure. It shows if a connection with the database has been established successfully. A confirmation or error message appears depending on the connection status.

Click **OK**.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Connect To A Microsoft SQL Server Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Microsoft SQL Server database can be used as a dynamic data source for [label objects](#) or [form objects](#).

To manually connect an object to a Microsoft SQL Server database, complete the following steps:

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Microsoft SQL Server** as the preferred database type.

Step 2: Connection information

This step defines the database file connection details.

Server defines the database server to be used for the connection. The available servers are listed automatically. To add a non-listed server, insert its name or location manually.

Authentication selects the user authentication type.

- **Use Windows authentication** to login using your Windows domain credential.
- **Use SQL Server authentication** to login using the SQL server credentials.

Database selection selects the database on the previously selected server. This database is going to be used as a data source for the selected label or form object.

- **Database** defines the server database to connect to.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard dialog inputs in the dialog box.

Test Connection button starts a connection testing procedure. It shows if connection with the database has been established successfully. A confirmation or error message appears depending on the connection status.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Connect To An Oracle Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Oracle database can be used as a dynamic data source for [label objects](#) or [form objects](#).

To manually connect an object to an Oracle database, complete the following steps:

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Oracle** as the preferred database type.

Step 2: Connection information

This step defines the database file connection details.

Server defines the database server to be used for the connection. The available servers are listed automatically. To add a non-listed server, insert its name or location manually.

Authentication selects the user authentication type.

- **Use Windows authentication** to login using your Windows domain credential.
- **Use SQL Server authentication** to login using the SQL server credentials.

Database selection selects the database on the previously selected server. This database is going to be used as a data source for the selected label or form object.

- **Database** defines the server database to connect to.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard dialog inputs in the dialog box.

Test Connection button starts a connection testing procedure. It shows if connection with the database has been established successfully. A confirmation or error message appears depending on the connection status.

Click **OK**.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Connect To MySQL Database

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

MySQL database can be used as a dynamic data source for [label objects](#) or [form objects](#).

To manually connect an object to a MySQL database, complete the following steps:

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **MySQL** as the preferred database type.

Step 2: Connection information

This step defines the database file connection details.

Database defines the exact database on a server. Enter the correct name.

Host defines the database server IP address or name.

Port defines the port of the database server.

Authentication provides user name and password for establishing the connection.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Connect To Other Databases (OLE DB)

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Various types of databases can be connected to [label objects](#) or [form objects](#) via OLE DB source.

OLE DB extracts data from a variety of OLE DB-compliant relational databases by using a database table, a view, or an SQL command.

EXAMPLE OLE DB can extract data from tables in Microsoft Access or SQL Server databases.

To manually connect an object to other databases via OLE DB, complete the following steps:

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **Other Databases (OLE DB)** as the preferred database type.

Step 2: Connection information

This step defines the OLE DB connection details.

Provider defines the provider to be used for accessing the data by exposing the OLE DB interfaces.

Authentication provides user name and password for establishing the connection.

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Authentication provides user name and password that are used for the connection.

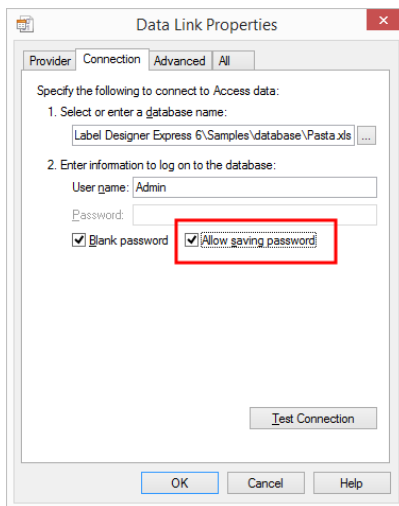
Advanced Configuration options are:

- **Automatically browse for tables** (selected by default) automatically displays the available OLE database tables. Cleared checkbox skips this step.

Show Connection String displays the current database connection string and allows it to be inserted or modified.

WARNING Connection string editing is intended for advanced users only. To configure the database connection, users are encouraged to use standard inputs or **Advanced Setup** dialog.

Advanced Setup button opens the *Data Link Properties* window allowing the user to define the connection properties. **Data Link Properties** is a Windows system dialog – read more about its properties [here](#).



NOTE When connecting to a password protected database, make sure the **Allow saving password** option is selected. If not, even after a successful **Test Connection** procedure, database access is not going to be granted.

Test Connection button starts a connection testing procedure to confirm if a connection with database has been established successfully. A confirmation or error message appears.

Click **OK**.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Connect To ODBC Data Source

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

Various databases can be connected to [label objects](#) or [form objects](#) via [ODBC Data Source Administrator](#).

To manually connect an object to a database, using the ODBC, complete the following steps:

Step 1: Select Database Connection

Open the [Dynamic Data Manager](#). This dialog enables the user to [manage the variable data sources](#) for label and form objects.

Click **Database Connections** button in the [Dynamic Data Manager ribbon](#) and select **ODBC Data Source** as the preferred database type.

Step 2: Connection information

This step defines the ODBC connection details.

Data Source defines the source to retrieve the data from.

Driver displays the database driver according to the selected data source.

ODBC Administrator button opens the system ODBC administration dialog. Read more details about the dialog [here](#).

Test Connection button starts a connection testing procedure. It checks if the Designer can successfully connect to the database or not.

Click **OK**.

Step 3: Configure Database Connection

Read how to set database connection details files [here](#).

Step 4: Set Table Configuration Details

Read how to configure the connected table [here](#).

Step 5: Define Database field Details

Read how to configure the connected table [here](#).

Click **OK**. The database is connected and ready to be used.

Add Or Remove Database Tables

Existing database connections are configurable at any time. To add or remove tables from a connected database file, open the [Dynamic Data Manager](#) and click the database in the [data source explorer](#).

Database Connection group gives information on the connection database.

- **Connection name:** defines the name for the connected database file.
- **Description:** allows adding additional information and suggestions for the connected database.
- **Connection identifies:** currently connected database file. To replace the currently connected file, click the Connection Setup button. New Database Connection Properties window reappears – repeat step 1 to connect to an alternative database file.

NOTE You can add the same database table more than once if different record filtering or sorting is required.

Tables group displays the available database tables and the tables that are selected to be used.

- **Available tables:** frame with available tables of the connected database. Select the tables from the list.
- **Selected tables:** tables to be further used as the data source.

Tables can be added to or removed from the **Selected tables** using **Add >** and **< Remove** buttons. To use the entire range of available tables, use **Add all >>** and **<< Remove All** buttons.

- **Refresh Tables:** rereads the connected database file and displays the refreshed available tables.

NOTE Tables that are already used as any object's data source cannot be removed. A warning appears when trying to remove such table.

Table Configuration

Table Configuration group allows you to configure the connected database table. Use the tabs below to browse through various configuration options.

- **Table name:** displays the selected database table's name.
- **Table alias:** gives a unique display name to a table. Table alias is useful when the same table is added for more than once under the same database connection. Alias identifies these tables when used in the Designer.

Fields Tab

Available Fields frame lists the available fields of the connected database table. Select the fields from the list.

Selected fields displays the fields to be further used as the data source.

The fields can be added to or removed from the **Selected fields** using **Add >** and **< Remove** buttons. To use the entire range of available fields, use **Add all >>** and **<< Remove All** buttons.

Refresh Fields rereads the connected database table and displays the refreshed available fields.

Filter Tab

Enable filter commands activates the table filter. Use it to filter out the displayed database fields as defined by a condition or a group of conditions.

Add condition button creates a custom filter. Select standard qualifiers: equals, does not equal, is less than, is less than or equal to, is greater than, is greater than or equal, like, not like, is blank, is not blank.

Add group button activates nesting two or more conditions for a filter. Use a group to build a more complex filtering condition for a field. The conditions can be joined using AND (all conditions must be true in order to display the record) or OR (only one condition must be true in order to display the record) logical qualifiers.

The list of defined conditions and groups is placed below the table. Remove the filter(s) by clicking the **Remove** button.

Sorting Tab

Field column shows the order in which the database fields are displayed in a table. To change the order, select a field and click **Move up** or **Move down** to place it at a desired position.

Sort Order defines whether the records as displayed **Ascending** or **Descending**. Select the sort order from the drop-down menu.

Data Retrieving Tab

Data selection at print time initialization group defines database print time record selection and printing options.

- **Show record selection at print time:** enables manual selection of database records before printing. The content of selected records is displayed in label objects and printed.

When enabled, this option adds a selection column to the database table on the print dialog. This column allows individual selection of the records to be printed.

- **Default print:** defines which database records would be selected in the print dialog by default.
 - **All records:** prints out the entire range of selected records.
 - **First record:** only prints out the first record in a table.
 - **Last record:** only prints out the last record in a table.

Number of copies per record sets print quantities for individual database records.

- **Copies per record:** defines how many labels should be printed per single record.
- **Number of copies can be changed at print time:** allows setting the number of printed label copies for a single database record. When enabled, this option adds a column to the database table. This column allows individual setting of print quantity for the selected record.

Advanced options allow you to set how multiple records can be displayed.

- **Collect records:** displays the content of multiple records in a single object.
 - **Delimiter:** defines how the records are separated when displayed in an object. Set New line (**CR/LF**) or select a special character from the list.
- **Limit number of collected records:** enables the maximum number of displayed records in a single object.
 - **Records:** sets the maximum number of records to be displayed in an object.
 - **Span multiple labels:** enables the records to be displayed in an object over multiple labels.
- **Use the same record for entire print job:** prints out the selected record only.

SQL Tab

SQL tab allows creating SQL statements. Commands in SQL statements determine how to obtain the data from the database. When creating a filter, the SQL sentence is auto-generated. Modify it or write your own sentence.

- **Edit SQL:** [converts table object into a query object](#). This button allows defining custom tables that are based on SQL queries.

NOTE This option is for experienced users only. If you make a mistake and create an invalid SQL statement, the query results become unpredictable. No data will be returned from the database or connection to the database will become impossible.

- **Export:** saves the current SQL statement as an SQL file on a disk.
- **Import:** allows external SQL statements to be used with the current database.

Data Tab

Data tab displays the connected database file table. Use search field and field selector to find the records.

Field Configuration

Details group allows defining the connected database field properties. Set these properties to make the use of a database as simple and efficient as possible.

- **Field name:** defined automatically by the source database file.
- **Field alias:** gives a unique display name to a field.

WARNING When using Python or Visual Basic Script, use field alias names that contain alphanumeric and underscore characters only. The names must not start with a digit.

- **Type:** identifies the data type of a database field. This property depends on the connected database field and cannot be edited.
- **Length:** (not available for [Text File](#)) displays the field length as defined by the database.
- **Code page:** provides support for the character sets used in different countries or regions. Code pages are referred to by number – select the appropriate one from the drop down list.

Data sets the limitations for the field length.

- **Limit field length (truncate excessive content):** enables the maximum field length limitation. Extra characters are removed.
 - **Length (characters):** defines the exact maximum field length using the number of allowed characters.

Databases With Custom SQL Queries

Designer allows defining custom tables that are based on SQL queries. Two methods are available for creating a custom SQL query:

1. Create a new SQL query.

Go to **Dynamic Data Manager** -> [Database Connection](#) **Tables** and click **Create new query** in the **Available tables** field.

2. Convert an existing database table into an query object.

Go to **Dynamic Data Manager** -> **Table Configuration** -> **SQL Tab** and click the **Edit SQL** button. This converts the connected database into a Query object.

NOTE This option is for experienced users only. If you make a mistake and create an invalid SQL statement, the query results become unpredictable. No data will be returned from the database or connection to the database will become impossible.

NOTE This option is not available for text database files.

Insert a custom query into the edit field. Click **OK** when done.

NOTE SQL statement field must not be left empty. An error appears if trying to continue without defining the statement.

Using Text File Structure Wizard

A "real" database must contain structured data. Text databases lack data structure, which means that the structure must be defined before a text file can be used as data source. Define the structure using the *Text File Structure Wizard*.

NOTE **Text File Structure Wizard** opens if a text file you are connecting to has not been previously used as object data source.

To complete the text file structure wizard, complete the below described steps.

Step 1: Welcome

Welcome window displays the text file you are going to convert into a database and use as a data source of an object. Make sure the correct text file is displayed under **Selected text file**.

Click **Next**.

Step 2: Data Encoding

This step sets the **Encoding** type. The following types are available:

- Auto
- ASCII
- UTF-8
- UTF-16
- UTF-16BE

When in doubt which encoding should be used, select **Auto** for automatic detection of encoding type. **Auto** identifies the encoding type by reading the BOM unicode character. If BOM is not included or is misinterpreted, **Auto** presumes, the text uses ASCII encoding. Inadequate character type identification might cause the database structure to be displayed incorrectly.

NOTE While selecting the encoding type, check the preview field. Correct values must be displayed.

Click **Next**.

Step 3: Data Structure

This step defines the fields to be used in the text database. There are two options:

- **Delimited:** fields are separated by a delimiter.
- **Fixed width:** fields with predefined (fixed) length.
- **First row contains field names:** defines if the field names are included in the first row of the database file.
 - **Start import at row** defines the row in the database file from which the data import starts. This option enables skipping the rows that do not include data.

Check the preview field. Click **Next** if the text content is displayed properly.

Step 4: Set Column Breaks

This step depends on the previously selected data structure option – **Delimited** or **Fixed width**.

Delimited opens the *Fields Delimiter* window.

- **Delimiter:** defines the delimiting character. Select among the standard characters or insert a custom delimiter in **Other** field.
- **Text qualifier:** character that indicates textual content. Text qualifier should be used if a delimiter is a part of the text field content. Text qualifier should be used to enclose such field – the text between two text qualifiers is treated as a single field although it contains a delimiter.

Fixed width opens the *Set Column Breaks* window. Use mouse pointer to place the vertical lines where the data fields are going to be separated. The lines indicate where new fields start.

Click **Next**.

Step 5: Fields

Fields window allows you to manipulate and fine-tune the field names and the order in which they are displayed. The below listed settings are also available:

- In case of **Delimited** fields, the **Field Name** can be customized.
- With **Fixed width** fields, the following settings are allowed:
 - **Include:** includes a field in the selection.
 - **Field name:** custom name for the field.
 - **Offset:** separation line distance from the left table edge.
 - **Length:** field length.

Click **Finish**. Text file database structure is set.

Internal Variables As Dynamic Data Source

PRODUCT LEVEL INFO This segment is applicable to NiceLabel Designer Pro and NiceLabel PowerForms.

An internal variable performs as a dynamic data source which holds a value that is automatically retrieved from a running application and system environment.

Select internal variables by clicking the **Internal Variables** button in the [Data Sources ribbon](#). Select the appropriate variable check boxes.

NOTE The variables in this set can neither be edited nor modified. Their value is updated with every printed label.

List of available internal variables with description:

LabelFileName	The path and file name of the currently opened label file.
ShortLabelName	The file name of the currently opened label file.
RequestedQuantity	The quantity of labels requested for printing. This is the number of labels printed.
TotalQuantityPrinted	Total quantity of the printed labels. The number is the sum of label quantities from all label batches.
CurrentBatchQuantity	The number of labels reached in the current label batch. The value is reset at beginning of each label batch in the printing process.
LabelPrinterName	The name of the printer driver currently selected for printing.
DefaultPrinterName	The name of the default printer driver.
UserName	The application username of the currently logged-in user. It will have a value only if in-application authentication is enabled.
SystemUserName	The Windows user name of the currently logged-in user.
ComputerName	The name of the computer on which the application is running.
SolutionFileName	Current solution file name.
ShortSolutionFileName	Current short solution file name.
SolutionFilePath	Path to solution file name.
FormName	The path and name of the form application used for label printing instead of Print dialog box.
ShortFormName	Short name of the form application used for label printing instead of Print dialog box.
EPCData	EPCData as read from the RFID tag.
LabelRevision	Label Revision Description.

Global Variables As Dynamic Data Source

PRODUCT LEVEL INFO the use of Control Center is applicable to LMS Pro and LMS Enterprise only.

Global variable is a type of variable that can be shared among multiple labels. Once defined, it is stored outside the current label.

The global variable's last value is stored after each confirmation and with each print action. The stored values are useful when continued numbering from preceding print jobs is required. Global variable values are stored in a separate file on a disk or on a Control Center.

Global variables are created manually in [Dynamic Data Manager](#) or using a Control Center.

- [Add and manage global variables.](#)
- [Configure global variables.](#)

When creating a copy of the label file that uses global variables and using it on another computer, make sure the global variable source is accessible (file or Control Center).

NOTE If you skip this step, the labeling application won't find the corresponding global variable. A warning message will appear.

TIP: All label or solution global variables are managed in [Data Source Explorer](#).

Adding Global Variables As Object Data Sources

To add a new global variable in the [Dynamic Data Manager](#), use one of the following methods:

- Click **Global Variable** button in the dialog ribbon. Global variable configuration window appears.
- Click **Add new global variable** under **Global variables** in [Data Source Explorer](#). Global variable configuration window appears.

TIP: A new global variable is listed in the toolbar and ready to be used as a dynamic object content source. Add an object to the design surface and assign the global variable to it.

Global Variable

Global variable is a type of variable that can be shared among multiple labels. Once it is defined, it is stored outside the current label.

NOTE If a global variable is not defined or inaccessible, a warning appears on the top of the dialog window. To create a global variable, click the link inside the warning. Make sure the correct data source is defined in the Options dialog.

General Tab

About group of settings identifies the global variable and sets its definition.

- **Persistent ID:** identifier of the global variable. It serves as a unique reference from any connected source. Allowed values are 10000–99999.
- **Name:** unique global variable name. This name is used as a user-friendly identifier.

NOTE Avoid using non-alphanumeric characters when defining the variable name.

Enter the name to make the variable easy to find when listed among other variables in the data source explorer.

- **Description:** is a field that allows adding additional information and suggestions.
- **Current value:** value that is assigned to a global variable when created. It is defined using one of the following methods:
 - Manually entering a fixed value. Characters from any [group of allowed characters](#) are permitted.
 - Using a [special character](#):
 - Special character can be entered manually using the less than/greater than signs, e.g. <CR>, <LF> ...
 - Special character can be selected from the drop down [list](#).

Make sure the inserted current value meets the criteria defined with **Output Rules** for each data type.

Counter group settings allow you to configure global variables that perform the role of a counter.

- **Do not use a counter:** prevents the global variable from being used as a label counter.
- **Incremental counter:** counter value increases along with the printed labels.
- **Decremental counter:** counter variable value decreases along with the printed labels.
 - **Step:** amount of units that represent the next state of counter value.
 - **Repetition:** number of repetitions for each counter value.

Input Rules Tab

Data defines the counter input criteria.

- **Allowed characters:** permitted characters for variable values. Groups of allowed characters for data input filtering are described in section [Groups of Allowed Characters](#).

EXAMPLE Non-numeric characters can also be used as counter values. **Alphanumeric** sets the sequence with Step = 3 and Initial value = 1 as 1, 4, 7, A, D, G, J, M, P, S, V, Y, b, e, h, ...

- **Limit length:** maximum length of a variable value.
 - **Length (characters):** specifies the exact permitted number of characters.
- **Fixed length:** variable must contain the exact given number of characters as defined in the **Limit variable length**.

Output Rules Tab

Prefix and Suffix are characters that are added to a variable value.

- **Prefix:** text placed in front of the variable value.
- **Suffix:** text placed behind the variable value.

Pad Character fills empty character position until the maximum variable length for a variable is reached. Pad character is enabled only if the **Limit variable length** in the Input rules tab is enabled.

- **Padding:** defines the mode of padding.
 - **Not used:** does not use padding.
 - **On left:** adds pad characters on the left side of the data value.
 - **On right:** adds pad characters on the right side of the data value.
 - **Surrounding value:** adds pad characters on both sides of the data value.
- **Character:** character used for padding.

Groups Of Permitted Input Characters

There are multiple variable format that may be used to filter the input. This helps avoiding mistakes when entering data. The user is only allowed to enter the permitted characters.

All	Select this format when there is no need to limit the variable input data. For example: a variable can be used to define changes in barcode, text and graphics.
Numeric	Use this format for numeric variables such as serial numbers, EAN and UPC barcodes. Only numeric characters in the range from 0 to 9 can be entered. Sequence: 0123456789
Alpha-numeric	Use this format when numbers and characters are mixed in the same variable. Characters from 0 to 9 and from A to Z can be entered. Sequence: 0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZ-abcdefghijklmnopqrstuvwxyz
Letters	Use this format for variables that only contain letters. Sequence: ABCDEFGHIJKLMNPOQRSTUVWXYZ-abcdefghijklmnopqrstuvwxyz
7-bit ASCII format	The variable will contain only characters with ASCII code from 0 to 127.
Hex	Use this format to allow entering hexadecimal numbers. Sequence: 0123456789ABCDEF
Custom	Use this format to customize the range of allowed characters.
Code 39, Code 128A, Code 128B, Code 128C, Code 128, Codabar	Use these formats to only permit the use of characters that are included in the corresponding barcode standards.

Forms and Solutions

Solution

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

NiceLabel Designer solution is a single label printing file that includes multiple standalone items or interconnected labels and/or forms.

A solution enables adding any number of labels, forms and common [variable data sources](#). By doing this, a single Designer solution file serves as a container that envelops multiple labels and forms.

How do labels and forms cooperate in a solution? A label alone can be designed, printed, and, if necessary, reprinted. Multiplied manual printing of a single label file is time consuming and difficult if the content needs to be constantly updated. Therefore, NiceLabel introduced the ability to create forms which are combined with labels in a complete printing solution file.

In a solution, labels specify the layout of printed labels. Forms make sure the content of printed labels is easily defined, edited, updated, and reprinted. Forms also offer the user the control over a wide range of data- and print-related actions.

The advantages of keeping multiple labels and forms in a single file are:

- simplified management of printing solutions
- simpler and time efficient label designing and printing
- simplified use of variable data sources

Read about how to create or edit a solution [here](#).

Label Setup Wizard

Label Setup Wizard guides you through the process of creating a new label. The wizard consists of four configuration steps and a summary:

- [Step 1: Select a Printer](#)
- [Step 2: Page Size](#)
- [Step 3: Label Layout](#)
- [Step 4: Label Dimensions](#)
- [Step 5: Summary](#)

After finishing these steps, the label is ready for editing and printing.

NOTE To quit the Label Setup Wizard during any step, press escape. The new label properties are set to default.

Import And Export

Import and Export group allows importing, publishing and exporting the solution files.

- **Import into Solution:** locates the label or solution files and imports them into the active solution. After clicking the **Import into Solution**, an open file dialog opens. Browse for the file to be imported and click **Open**.

- **Export Label:** saves the label to disk and makes it available for use in another solution. After clicking **Export Label** the Export label dialog appears. Select a location to save the label to.

Form

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

NiceLabel Designer form serves as a panel for entering, viewing and selecting the data to be presented and printed on a label. The advantage of using a form are simplified data-entry and label printing process for the end-user.

In NiceLabel Designer, a form is created within a printing solution. This means that a form is usually built in combination with a predesigned label.

Read about how to create, design or edit a form [here](#).

Form Properties (Form Manager)

Form Manager

Basic Settings

Basic Settings tab is used to define the title, size and startup behavior of a form.

Title defines the form ID.

- **Show form title bar:** window title bar visible or hidden upon form startup.
- **Allow closing form:** form close using the window **Close** button allowed or not.

With this option disabled, the form can be closed from taskbar.

- **Allow resizing form:** form size customizable or not.

Disable this option to lock the form size.

Size group defines the form's **Width** and **Height**.

Initial form state group defines the form state upon startup.

- **Maximized:** form opens in full screen mode.
- **Default form size:** the form appears using the manually defined sizes.

Startup form position group defines the on-screen position of a form upon its startup.

- **As defined:** the form appears at a location defined by the distance in pixels from **Left** (left edge of the form) and **Top** (top edge of the form).
- **Screen center:** screen center is the startup form position.

Additional Settings

Additional Settings tab allows selecting the form scripting language. There are two scripting languages available for Designer form objects: **VBScript** and **Python**.

- **VBScript:** scripting for advanced data operations, comparisons and direct calculations on a form.

- **Python:** suitable for 64-bit systems. A significantly faster scripting alternative to **VBScript**.

Style

Set the label style parameters.

- **Background color** is defined by the **Standard** or **Advanced** color selection. Switch between these two options by clicking the **Advanced** or **Basic** button.
- Browse for the label **Background picture** or insert the direct path. Once the picture has been defined, it is possible to:
 - **Embed the picture to document:** makes the picture an integral part of label document.
 - **Save embedded picture to file:** embedded picture is saved to a file.
 - **Picture position:** background picture to be centered, to fit the label dimensions, or to be stretched.

Tab Order

Tab order tab customizes the order of setting the focus on form objects while pressing the **Tab** key.

- **ENTER key behaves as TAB key:** **Enter** key has the same role as the **Tab** key does.
- Select the form **Object** and move it up or down to define the focus switching order.

F1 Help

F1 Help tab defines custom form help content to help the end-user while designing and/or using a form.

TIP: Enter custom text in the editing field and click **OK**.

Events

Events tab allows setting actions for basic form events.

- **On Form Load:** the action is run upon form load.
- **On Form Close:** the action is run when the form is closed.
- **On Form Timer:** the action is run after a given time interval.
 - **Interval:** duration of the time interval.

TIP: Click [Actions ...](#) to set the actions that are run by the listed events.

Variable Events

Variable Events selects the variables that are monitored for changes in their values.

- **Add:** adds a [variable](#) to the list.
- **Delete:** removes a [variable](#) from the list.

TIP: Click [Actions ...](#) to set the actions that are triggered by changed values in the listed variables.

Info

Info tab includes a **Description** that serves as a hint or as a guidance for the user that is going to work with the form.

Define form **Description** by entering text into the field.

Form Objects

Frame

Frame object is used for creating rectangle shaped areas on a form. It visually separates multiple form fields.

Style

Style tab defines visual appearance of an object:

- **Background color:** object background color.
- **Transparent:** transparent frame.
- **Show border:** frame border show/hide.
- **Border color:** object border color selection.
- **Border width:** border width definition.
- **Border style** selects the object border style:
 - **None:** invisible border.
 - **Lowered:** the object appears lower than form surface.
 - **Raised:** the object appears higher than form surface.
 - **Lowered border:** the border of an object appears lower than form surface.
 - **Raised border:** makes the border of an object appear higher than form surface.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.

- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint (tooltip) helps the form users by briefly explaining why or how to use an object. Hint is shown to a user when the mouse pointer floats over the selected object.

Initial state on form startup group defines the object behavior when the form is run for the first time:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Text

Text is a form object for inserting textual content.

Content

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- **Variable keyboard input:** type of variable that enables the content of a prompted field to be different for every print job.
- **Variables:** predefined variable values which are used as object content.
- **Functions:** input data transformation tools.
- **Databases:** database values which are used as object content.

- [Link to other objects](#): makes the content of a label object (re)appear in another object on the same label.

Content field is used for entering the object content.

Settings

Text Settings tab defines if the object size or text should adapt to the amount of entered content.

- **Auto size**: automatically adapts the object size to the size of entered text.
- **Word wrap**: wraps the text to make it fit into the text box.

TIP: If the text box size is too small, a vertical scroll bar appears.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent**: transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left**: text aligned with the left object border.
- **Center**: text positioned centrally.
- **Right**: text aligned with the right object border.
- **Justified**: object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y**: anchoring point coordinates.
- **Width** and **Height**: horizontal and vertical object dimension.
- **Keep aspect ratio**: simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form**: object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form**: object width adapts to the resized form.
 - **Vertically resize with form**: object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Mouse Enter**: action is run on mouse enter.
- **On Mouse Leave**: action is run on mouse leave.
- **On Click**: action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint (tooltip) helps the form users by briefly explaining why or how to use an object. Hint is shown to a user when the mouse pointer floats over the selected object.

Initial state on form startup group defines the object behavior when the form is run for the first time:

- **Enabled**: defines if the object is going to be active (editable) at form startup or not.
- **Visible**: defines if the selected object is going to appear on the form or not.
 - **Condition**: an object is enabled and/or visible if the result of the given condition is "True".

Picture

Picture is a form object for inserting graphic content. The following file formats are supported:

- Enhanced Windows Metafile (.emf)
- Windows Metafile (.wmf, .wmz, .emz)
- Portable Network Graphic (.png)
- TIFF bitmaps (.tiff)
- JPEG bitmaps (.jpg)
- PDF
- Windows bitmap (.bmp)

- Paint
- Adobe Photoshop

Content

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- **Variables:** predefined variable values which are used as object content.
- **Functions:** input data transformation tools.
- **Databases:** database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Content field is used for entering the object content.

To (re)define the object **Content**, click **Browse** and locate the file to be displayed on the label.

Embed picture in a document defines the picture as an integral part of the label file.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Graphic Resizing tab defines variable source picture resizing.

Resize options group defines how the source file dimensions adapt to the size of Picture object at print time.

NOTE Resize options available only if the Picture object is connected to a variable data source.

- **Keep original picture size:** disables resizing. The source picture file is displayed in Picture object with its original dimensions.
- **Resize proportionally:** makes the source picture file resize proportionally. The aspect ratio of source file dimensions is preserved.
- **Resize to the designed size:** resizes the source picture file horizontally and vertically to make it fit into the bounding box. Using this option will most likely distort the image.

Original size group informs the user about the size of source image file.

Revert to original picture size resizes the Picture object to the original dimensions of source image file.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Mouse Enter:** action is run on mouse enter.
- **On Mouse Leave:** action is run on mouse leave.
- **On Click:** action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint (tooltip) helps the form users by briefly explaining why or how to use an object. Hint is shown to a user when the mouse pointer floats over the selected object.

Initial state on form startup group defines the object behavior when the form is run for the first time:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Button

Button adds a clickable and customizable object to a form. Its role is to activate various actions.

Source

Connected data source is the dynamic data source that is connected with the object.

- **Fixed data:** manually entered fixed text.
- **Variables:** predefined variable values which are used as object content.
- **Functions:** input data transformation tools.
- **Databases:** database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Content field is used for entering the object content.

Settings

Keyboard shortcut defines any keyboard to act as a shortcut. If the defined keyboard key is pressed, it acts as if the user would use a mouse click for running an action.

Default form button invokes the assigned action when a user presses Enter.

Only one button is allowed to be defined as the default form button.

Word wrap divides the text into multiple lines. It makes sure the text is not wider than the button.

Use a picture on the button group defines a graphic file to be displayed on a button.

- **Picture file name:** graphic file selected to be used on a button.
- **Embed picture in a document:** picture embedded in the document.

Whenever an embedded picture is needed, it is retrieved from the document and not from the file system.

- **Save embedded picture to file:** embedded picture is saved to a file.

If the picture is embedded, this action enables saving it at a selected location. The picture is no longer embedded.

- **Picture position:** picture position in relation to the object text.
- **Force original size:** full-size graphic without resizing is used on a button.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.

- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Click:** action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint (tooltip) helps the form users by briefly explaining why or how to use an object. Hint is shown to a user when the mouse pointer floats over the selected object.

Initial state on form startup group defines the object behavior when the form is run for the first time:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Edit Field

Edit field object is used for inserting and editing a single line of text.

Source

Connected data source is the dynamic data source that is connected with the object.

- [Variables:](#) predefined variable values which are used as object content.
- [Databases:](#) database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Settings

Settings tab contains two editable properties:

- **Automatically move focus to next control:** the next defined object on a form is automatically active after inserting a value. The Edit Field must be connected to a data source and must have a limited length defined.
- **Password field:** makes the in Edit Field characters invisible. The characters are masked with asterisks.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Change:** action is run when a change in the Edit Field object occurs.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Read-only:** prevented input and content editing.

- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Content after a print action group defines how the object content is handled after each printout.

- **Reset content after print:** object content reset after printing.
 - **Clear content:** object emptied after printing.
 - **Reset to initial content:** content reset after printing to the initially defined object content.

Memo Field

Memo field object is used for inserting textual content in multiple lines.

Source

Connected data source is the dynamic data source that is connected with the object.

- [Variables](#): predefined variable values which are used as object content.
- [Databases](#): database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Settings

Settings group contains two editable properties.

- **Automatically move focus to next control** makes the next defined object on a form automatically active after inserting a value.

TIP: The Edit Field must be connected to a data source and must have a limited length defined.

- **Password field** option makes the characters in this edit field invisible. They are masked with asterisks.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.

- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Change:** action is run when a change in the Edit Field object occurs.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Read-only:** prevented input and content editing.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Content after a print action group defines how the object content is handled after each printout.

- **Reset content after print:** object content reset after printing.
 - **Clear content:** object emptied after printing.
 - **Reset to initial content:** content reset after printing to the initially defined object content.

Combo Box

Combo box is used as an object for user input. Its role is to let the user select an option from a drop-down list or to add a custom value to the list.

Source

Connected data source is the dynamic data source that is connected with the object.

- [Variables](#): predefined variable values which are used as object content.
- [Databases](#): database values which are used as object content.

Settings

Settings tab defines object content editing specifics and displaying of values.

Allow input in run mode enables entering custom values when the form is running.

Allow duplicates allows duplicated values appear on the drop down list.

Is sorted sorts the list elements in ascending order. **Use case sensitive sort** additionally determines if the letter case should affect the sorting order or not.

Values group of settings allows defining the listed elements:

- **Items source:** defines the source for listed items.
 - **Custom values:** static user-defined values.
 - **Installed printers:** list of installed printers.
 - **Database field:** retrieved values from a connected database.
 - **Field:** selection of connected database field to retrieve the content from.
 - **Use another field for connected data source option:** connects another database field to the connected data source. The **Values content list** still displays the **Field** values, but the connected data source receives the selected value from a field in **Value field**.

EXAMPLE

1. Combo/List box object details

Items source:

Description	Product ID
CASONCELLI ALLA CARNE 250G	CAS006
BIGOLI 250G	PAS501
TAGLIATELLE 250G	PAS502GI
TAGLIOLINI 250G	PAS503GI

Database field: Description

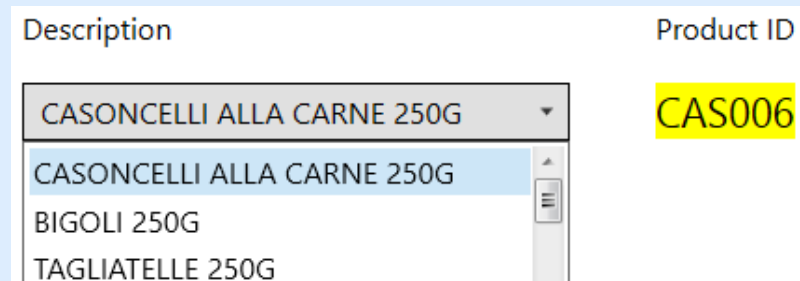
Connected data source: Variable1

2. Text object details

Added Text object should only display matching Product ID values.

Connected data source: Variable1

Result: The selected Description in Combo/List box results in the matching Product ID in Text object.



- **Value field:** selects the database field that is sent to the object's **Connected data source** and displayed as its content.
- **File names:** lists all files in the selected directory.
 - **Directory:** defines the path from where the labels are going to be listed.
 - **File mask:** specifies the filter for selecting the listed files.

EXAMPLE

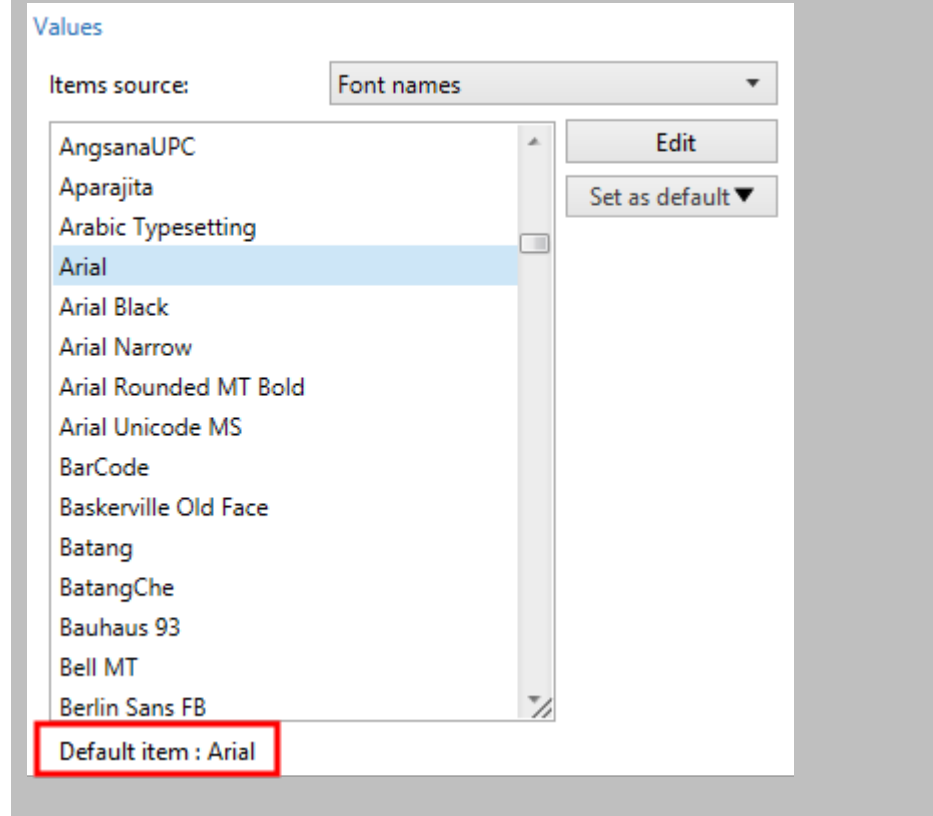
. lists all files

*.lbl lists only files with .lbl extension.

t*.lbl lists files that start with "t" and have the extension .lbl.

- **Show the file path:** entire file path is displayed on the list.
- **Show the file extension:** file extension visible on the list.
- **Font names:** lists the installed fonts.
- **Labels in solutions:** lists all labels within the solution.
- **Edit:** converts system-defined items on the **Values content list** into a list of custom values.
- **Values content list:** displays the current object content.
- **Set as default:** turns the currently active selection into a default value.

Default value is a value that is automatically selected when the form is run.



NOTE All values except for custom values are populated when the form is run. The values displayed at design time are sample values retrieved from the system. After clicking the **Edit**, Designer makes a snapshot of values and makes them editable in the **Custom Values** dialog.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Change:** action is run when a change in the Edit Field object occurs.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Read-only:** prevented input and content editing.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Content after a print action group defines how the object content is handled after each printout.

- **Reset content after print:** object content reset after printing.
 - **Clear content:** object emptied after printing.
 - **Reset to initial content:** content reset after printing to the initially defined object content.

List Box

List box is used as a user input object. Its role is to let the user select an option from a list.

TIP: Unlike [Combo box](#), List Box does not allow inserting custom values.

Source

Connected data source is the dynamic data source that is connected with the object.

- [Variables:](#) predefined variable values which are used as object content.
- [Databases:](#) database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Settings

Settings tab defines object content editing specifics and displaying of values.

Allow duplicates allows duplicated values appear on the drop down list.

Is sorted sorts the list elements in ascending order. **Use case sensitive sort** additionally determines if the letter case should affect the sorting order or not.

Values group of settings allows defining the listed elements:

- **Items source:** defines the source for listed items.
 - **Custom values:** static user-defined values.
 - **Installed printers:** list of installed printers.
 - **Database field:** retrieved values from a connected database.
 - **Field:** selection of connected database field to retrieve the content from.
 - **Use another field for connected data source option:** connects another database field to the connected data source. The **Values content list** still displays the **Field** values, but the connected data source receives the selected value from a field in **Value field**.

EXAMPLE**Field:** ObjectField1**Value field:** ObjectField2**Connected data source:** Variable1**Result:**

Object connected to Variable1 displays the content from ObjectField1 and sends the content from ObjectField2 to the Variable1.

- **Value field:** selects the database field that is sent to the object's **Connected data source** and displayed as its content.
- **File names:** lists all files in the selected directory.
 - **Directory:** defines the path from where the labels are going to be listed.
 - **File mask:** specifies the filter for selecting the listed files.

EXAMPLE

*. * lists all files

*.lbl lists only files with .lbl extension.

t*.lbl lists files that start with "t" and have the extension .lbl.

- **Show the file path:** entire file path is displayed on the list.
- **Show the file extension:** file extension visible on the list.
- **Font names:** lists the installed fonts.
- **Labels in solutions:** lists all labels within the solution.
- **Edit:** converts system-defined items on the **Values content list** into a list of custom values.
- **Values content list:** displays the current object content.
- **Set as default:** turns the currently active selection into a default value.

Default value is a value that is automatically selected when the form is run.

NOTE All values except for custom values are populated when the form is run. The values displayed at design time are sample values retrieved from the current computer.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.

- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Click:** action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Content after a print action group defines how the object content is handled after each printout.

- **Reset content after print:** object content reset after printing.
 - **Clear content:** object emptied after printing.
 - **Reset to initial content:** content reset after printing to the initially defined object content.

Radio Group

Use **Radio Group** object to allow a user to select a single item from a set of mutually exclusive items.

Source

Connected data source is the dynamic data source that is connected with the object.

- [Variables:](#) predefined variable values which are used as object content.
- [Databases:](#) database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Settings

Settings tab defines object content editing specifics and displaying of values.

Allow duplicates allows duplicated values appear on the drop down list.

Is sorted sorts the list elements in ascending order. **Use case sensitive sort** additionally determines if the letter case should affect the sorting order or not.

Values group of settings allows defining the listed elements:

- **Items source:** defines the source for listed items.
 - **Custom values:** static user-defined values.
 - **Installed printers:** list of installed printers.
 - **Database field:** retrieved values from a connected database.
 - **Field:** selection of connected database field to retrieve the content from.
 - **Use another field for connected data source option:** connects another database field to the connected data source. The **Values content list** still displays the **Field** values, but the connected data source receives the selected value from a field in **Value field**.

EXAMPLE

Field: ObjectField1

Value field: ObjectField2

Connected data source: Variable1

Result:

Object connected to Variable1 displays the content from ObjectField1 and sends the content from ObjectField2 to the Variable1.

- **Value field:** selects the database field that is sent to the object's **Connected data source** and displayed as its content.
- **File names:** lists all files in the selected directory.
 - **Directory:** defines the path from where the labels are going to be listed.
 - **File mask:** specifies the filter for selecting the listed files.

EXAMPLE

. lists all files

*.lbl lists only files with .lbl extension.

t*.lbl lists files that start with "t" and have the extension .lbl.

- **Show the file path:** entire file path is displayed on the list.
- **Show the file extension:** file extension visible on the list.
- **Font names:** lists the installed fonts.
- **Labels in solutions:** lists all labels within the solution.
- **Edit:** converts system-defined items on the **Values content list** into a list of custom values.
- **Values content list:** displays the current object content.
- **Set as default:** turns the currently active selection into a default value.

Default value is a value that is automatically selected when the form is run.

NOTE All values except for custom values are populated when the form is run. The values displayed at design time are sample values retrieved from the current computer.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Click:** action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint (tooltip) helps the form users by briefly explaining why or how to use an object. Hint is shown to a user when the mouse pointer floats over the selected object.

Initial state on form startup group defines the object behavior when the form is run for the first time:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Check Box

Check Box is a form object that allows the user to make a binary choice – select or deselect the listed options.

Source

Source tab defines data sources, data types, values, and prompting rules.

Connected data source is the dynamic data source that is connected with the object.

- [Variables](#): predefined variable values which are used as object content.
- [Databases](#): database values which are used as object content.

NOTE The down arrow object button provides direct access to [dynamic data sources](#). Click the arrow to add a new object on the design surface and to connect it with the selected data sources simultaneously.

Settings

Settings tab defines specifics for object content editing and displaying of values.

Check box text is a field for entering the Check Box text.

- **Checked:** default Check Box state (selected/cleared) when the form is run.
- **Word wrap:** text is divided into multiple lines to make sure it does exceed the Check Box width.

State values define resulting actions of the Check Box object :

- **Checked value:** by default set to "True". Checked option confirms an action.
- **Unchecked value** by default set to "False". Checked option rejects an action.

Checked and **Unchecked** values are customizable. These values can be defined manually or dynamically using a **Data source**.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.

- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Focus:** action is run when focus is set on the object.
- **On Exit:** action is run when focus moves to another object.
- **On Click:** action is run on mouse click.

General

General tab identifies the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) at form startup or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Content after a print action group defines how the object content is handled after each printout.

- **Reset content after print:** object content reset after printing.
 - **Clear content:** object emptied after printing.
 - **Reset to initial content:** content reset after printing to the initially defined object content.

Database Table

Database Table object is used for displaying a selected database table on a form. Such table allows searching, filtering and selecting the database records on a form.

Settings

Settings tab allows selecting the database table.

Table displays the currently used (active) database table.

TIP: Add a database by running the [Database wizard](#) or select it from the databases that have been defined using the [Dynamic Data Manager](#).

- **Enable multiple row selection** allows selecting multiple database records simultaneously.
 - **Enable selection with check box:** added selection check box in front of database records.

This option enhances the use of touch screen devices. Selection of multiple records becomes more user-friendly.

Show search controls option shows/hides the database search on the form.

Columns group allows managing the connected database table columns.

- **Style** button opens the [Column style dialog window](#). This dialog enables the user to customize the visual appearance of a selected table column or cell.
- **Move up** button places the selected record one position higher.
- **Move down** button places the selected record one position lower.
- **Field alias** column displays the name of the table field as defined in the source database.
- **Caption** allows defining a custom column name.
- **Width** defines the table column width.
- **Visible** makes the table column visible or hidden on the form.
- **Variable** defines the variable which stores the selected table value.

Style

Style tab defines visual appearance of the Database Object table.

- **Alignment:** alignment of table header row.
- **Background color:** table background color.
- **Font color:** table text font color.
- **Font:** table text typeface and its properties (**Bold, Italic** and **size**).

Cell style defines visual appearance of a cell in a database table.

- **Alignment:** table cell content alignment.
- **Background color:** cell background color.
- **Font color:** cell font and underline colors.
- **Font:** cell text typeface and its properties (**Bold, Italic** and **size**) to be used for the cell content.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing

a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Read-only:** prevented input and content editing.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Column Style Dialog

Column style dialog allows defining style-related table properties of the [Database Table](#) object.

Column style group defines visual appearance of the table column.

- **Override table default style** enables style customization of the selected column.
- **Alignment** defines the header row content alignment.
- **Background color** defines the column background color. **Transparent** makes the column background invisible.
- **Font color** specifies the font color.
- **Font** allows specifying the typeface and its properties: **Size**, **Bold** and **Italic**.

Cell style group defines visual appearance of individual cells.

- **Override table default style** enables style customization of the selected cell.
- **Background color** defines the cell background color. **Transparent** makes the cell background invisible.
- **Font color** specifies the font color.
- **Font** allows specifying the typeface and its properties: **Size**, **Bold** and **Italic**.

Database Navigator

Database Navigator object is used as a tool for navigating, adding and deleting the database records on a form.

Settings

Table defines the database for navigating.

TIP: Add a database by running the [Database wizard](#) or select it from the databases that have been defined using the [Dynamic Data Manager](#) dialog.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.

- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Read-only:** prevented input and content editing.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Database Search

Database Search object is a search tool for databases on a form. Search locates the record while typing the matching characters or selects it according to the database column.

Settings

Table defines the database to be searched.

TIP: Add a database by running the [Database wizard](#) or select it from the databases that have been defined using the [Dynamic Data Manager](#) dialog.

- **Search on every keypress (incremental search):** places the cursor on the first database record that contains the defined sequence of characters. Each

keypress repeats the procedure.

- **Select record only when exact match is found** exact match for finding a record is required.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the print form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Label Preview

Label Preview object is used for giving a live print preview with defined print parameters for the selected label.

Settings

Label specifies the label file whose preview is displayed in the object.

TIP: Click **New label** to create a new label within the solution. If the label is not supposed to be a part of the solution, locate it using the **Browse** button. The label can also be defined dynamically using a connected data source.

Printer defines the printer whose settings are used for generating the preview.

TIP: If no other printer is defined, the printer defined for the active label is used. The printer can also be defined dynamically using a connected data source.

Content group defines what the Label Preview includes:

- **Show a single label:** preview of label's printable area.
- **Show all labels on the page:** preview of the entire page containing the labels.

NOTE This option is useful when [Labels Across](#) is in use or when previewing the label margins.

Quantity group defines the number of previewed labels.

- **Labels:** the number of label to be displayed in the Label Preview.
- **All (unlimited quantity):** prints the entire range of labels depending on the data.

Number of skipped labels defines the numbers of labels to be skipped on the first page of preview.

TIP: This option is used with [Labels Across](#).

Identical copies per label defines the number of copies for each label in the preview.

Number of label sets specifies how many times the entire label preview should repeat.

Label Side defines the side of the label to be displayed in the preview.

- **Show front side:** front side of the label appears in the preview.
- **Show back side:** back side of the label (if available) appears in the preview.

NOTE If both options are selected, both label sides appear on the preview.

Style

Style tab defines visual appearance of an object.

- **Background color:** Label Preview background color.
- **Transparent:** transparent object background.
- **Show border:** Label Preview object border visible.
- **Border color:** Label Preview border color.
- **Border width:** Label Preview border width.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

Events

Events tab defines the actions that are run by various object-related events.

TIP: See section [Actions Editor](#) to read more about this powerful Designer tool.

Available events are:

- **On Mouse Enter:** action is run on mouse enter.
- **On Mouse Leave:** action is run on mouse leave.
- **On Click:** action is run on mouse click.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the print form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Data Initialization

Data Initialization object is used as a panel for assigning initial values to variables on the selected label.

Settings

Label selects the label that is going to be used with Data initialization object.

TIP: If the label is not a part of the solution, it can be located using the **Browse** button. The label can also be defined dynamically using a connected **Data source**.

Focus Data Initialization when label changes sets focus to the Data Initialization table when the selected label changes. This makes the table instantly editable.

Show selection of labels from the solution adds a drop down list that displays all the labels contained in the solution.

TIP: This drop down list allows the user to change the active label that is going to be printed. If this option remains unchecked, select the label by setting the **Label** data source.

Show database table initialization allows the user to select database records and to define the printed quantities for each of them.

NOTE A separate tab is added for each database table that is connected to a label.

Columns group sets the width of data initialization table and its **Prompt, Value** and **Formatted value** columns.

- **Auto-size:** automatic column resizing.
- **Show fomatted value:** visible **Formatted value** column.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the print form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing

a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Printer Settings

Printer Settings object enables adjusting printing speed and darkness on a form.

NOTE The object overrides the currently defined driver settings – printing within the active solution uses properties as defined using this object.

Settings

Label specifies the label file to be used with Printer Settings object.

TIP: If the label is not a part of the solution, it can be located using the **Browse** button. The label can also be defined dynamically using a connected **Data source**.

Printer defines the printer whose settings are used.

TIP: If no other printer is defined, the printer defined for the active label is used. The printer can also be defined dynamically using a connected **Data source**.

Show speed settings sets printing speed setting availability.

Show darkness settings sets printing darkness setting availability.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold**, **Italic**, **Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X** and **Y:** anchoring point coordinates.
- **Width** and **Height:** horizontal and vertical object dimension.

- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form:** object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form:** object width adapts to the resized form.
 - **Vertically resize with form:** object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the print form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled:** defines if the object is going to be active (editable) on the print form or not.
- **Visible:** defines if the selected object is going to appear on the form or not.
 - **Condition:** an object is enabled and/or visible if the result of the given condition is "True".

Print Quantity

Print Quantity object defines the number of labels (or pages of labels) to be printed.

Settings

Label specifies the label file to be used with Print Quantity object.

Print Quantity is defined using a connected variable value.

TIP: The variable must be used as the quantity that is defined for the print action.

Show additional settings group allows defining the following properties:

- **Number of skipped labels variable:** assigns the selected variable a number of labels to be skipped on first page.

NOTE When defining the number of skipped labels, duplicates, or label sets, a new window appears. This window allows the user to enter the values.

This option is used with [Labels Across](#).

- **Identical copies per label variable:** assigns the selected variable a number of copies for each label in a print job.
- **Number of label sets variable:** assigns the selected variable a value that specifies how many times the entire label printing process should repeat.

WARNING At least one variable must be defined when the **Show additional settings** option is enabled.

Style

Style tab defines visual appearance of an object.

Background color defines the object background color.

- **Transparent:** transparent object background.

Font color defines the font and underline colors.

Font selects the typeface.

The font may appear **Bold, Italic, Underlined** or as a **Strikethrough** text.

Alignment defines horizontal relative positioning of the entered content.

- **Left:** text aligned with the left object border.
- **Center:** text positioned centrally.
- **Right:** text aligned with the right object border.
- **Justified:** object text distributed equally to both sides.

Position

Position tab defines object positioning and its position-related behavior.

Position group defines the object position.

- **X and Y:** anchoring point coordinates.
- **Width and Height:** horizontal and vertical object dimension.
- **Keep aspect ratio:** simultaneous changing of object dimensions while scaling.
- **Lock object on design surface** prevents the object from being moved during the design process.

Size group sets the object's dimensions:

- **Resize anchor point** defines the fixed distance of an object from the form borders.

Choose the most appropriate anchor point to ensure the object's visibility regardless of the current window size.

- **Horizontally resize with form** and **Vertically resize with form**: object size automatically adapts to the changing size of the form.
 - **Horizontally resize with form**: object width adapts to the resized form.
 - **Vertically resize with form**: object height adapts to the resized form.

NOTE If both options are enabled, object width and height adapt to the resized form simultaneously.

Rotation angle is the object angle according to the design surface.

General

General tab defines the object and defines object settings for form startup.

Name sets a unique object ID. It is used for object referencing when defining functions, variables, scripts, etc.

Description allows adding notes and annotations for an object.

Hint helps the print form users by briefly explaining why or how to use the selected object.

Initial state on form setup group defines the object behavior while editing and printing a form:

- **Enabled**: defines if the object is going to be active (editable) on the print form or not.
- **Visible**: defines if the selected object is going to appear on the form or not.
 - **Condition**: an object is enabled and/or visible if the result of the given condition is "True".

Define Actions

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

Actions are an essential part of automated labeling solutions. Each action performs a pre-defined command (or a series of commands) when a defined event happens.

Designer includes a wide range of actions. Their purpose is to eliminate the need to start the solution programming from scratch.

Manage the actions using the [Actions Editor](#) dialog box.

Basic **Action** concepts and properties are described below.

- [Available Actions](#): the range of actions that are included in Designer. These actions are grouped into functional sets.
- **Defining actions**: an action is defined in **Actions Editor** by clicking the appropriate action icon in **Add** ribbon group. The main ribbon contains commonly used actions and – later – the actions you define as common actions. To see all available actions, click **All Actions**.
- **Nested Actions**: actions that cannot be used on their own. Their specific characteristics require them to be nested within another action. Use buttons in [Action Order ribbon](#) group to change action placement. Each action is identified with an ID number that indicates its position in the list, including its nesting. This ID num-

ber is displayed in the potential error message so you can find the problematic action faster.

[Print Label](#) action is an example of such action. This action is nested under the [Open Label](#) action, so it references the exact label to be printed.

- **Action Execution:** listed (active) actions are executed once per event. The order of actions is crucial – the execution begins at the top and moves toward bottom of the list.
- **Conditional Actions:** conditional actions only run when the provided conditions allow them to be run. Condition is defined with a single line [VBScript Expression](#) or [Python script](#).
- **Errors in Actions:** if an action is not configured completely, it is marked with a red exclamation icon. Such action can be included in the event list but cannot be executed.

NOTE If one of the nested actions reports an error, all parent actions are also colored red. This serves as an indication for a nested action error.

- **Disabling Actions:** prevents an action from being executed. By default, each added action is enabled. Actions that are not needed, can be disabled and still kept in the configuration. The shortcut to action enabling & disabling is the check box in front of the action name on the list of defined actions.
- **Copying Actions:** any action may be copied and pasted. Use standard Windows keyboard shortcuts, or right-click the action.

Actions Editor

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

Actions Editor is a dialog for managing [actions](#) in a Designer [solution](#). To open the dialog box, select a form object, press F4 to open the form properties and click **Actions...**

Ribbon

Actions Editor Dialog ribbon includes commands for adding, removing and ordering the actions. It also provides a direct access to frequently used actions.

Clipboard group icons activate the following actions:

- **Paste:** pastes the clipboard data.
- **Cut:** cuts the selection to the clipboard.
- **Copy:** copies the selection to the clipboard.
- **Delete:** deletes the selected items.

Undo & Redo group allow undoing or repeating actions.

- **Undo:** Designer allows the user to undo the entire sequence of actions since the last file save.
- **Redo:** repeats the requested range of actions.

Action Order group defines the action execution order of selected actions.

- **Up** and **Down**: arrows place the selected action in front or after any other existing action.
- **Right**: arrow nests the selected action under the previous existing action.

NOTE A nested action is any action that starts when the parent action is already in progress.

- **Left**: arrow makes a nested action independent of the preceding action.

NOTE Certain actions cannot exit independently. If such action is added to the action list, a warning appears. The warning defines which action should it be nested under.

Add assigns actions to the selected form object.

- **All actions** button gives access to the entire range of [Designer actions](#). **Recently used** actions are listed on the top. Use **Search...** field to quickly locate any action by entering its name.
- Four buttons give direct access to the most commonly used actions:
 - **Open Label**: button adds the [Open Label](#) action to the event list.
 - **Print Label**: button adds the [Print Label](#) action to the event list.
 - **Set Printer**: button adds the [Set Printer](#) action to the event list.
 - **Quit**: button adds the [Quit](#) action to the event list.

Actions Explorer

Actions Explorer is a tool for adding, removing and ordering the assigned actions. Use ribbon commands to manipulate with existing actions or to add new actions.

Editing Field

Editing field allows editing the advanced action properties.

- Main properties of the selected action are available for editing on the top of the Main/editing field. Main properties differ with each action – read the dedicated [action description sections](#) for details.
- Hidden properties define the less frequently defined properties. Hidden properties differ with each action – read the dedicated [action description sections](#) for details.

Available Actions

PRODUCT LEVEL INFO This section is applicable to NiceLabel PowerForms.

The Designer actions are grouped into multiple functional sets. The groups with basic action descriptions are listed below.

General group contains frequently used label opening and activation related commands:

- [Open Label](#)
- [Print Label](#)
- [Execute Script](#)
- [Open Document/Program](#)

Printer group contains actions related to printing:

- [Set Printer](#)
- [Define printer settings](#)
- [Set Print Job Name](#)
- [Redirect Printing to File](#)
- [Set Print Parameter](#)
- [Redirect Printing to PDF](#)
- [Printer Status](#)
- [Store Label to Printer](#)

Form group defines actions related to form objects:

- [Open Another Form](#)
- [Message](#)
- [Quit](#)
- [Move Focus](#)
- [Translate form](#)
- [Set Object Property](#)

Variables group defines actions related to variables:

- [Set Variable](#)
- [Save Variable Data](#)
- [Load Variable Data](#)
- [String Manipulation](#)

Data & Connectivity group defines the actions related to databases, data sending, reading or receiving, and networking.

- [Execute SQL Statement](#)
- [Refresh Table](#)
- [Import Data into Table](#)
- [Send Data to Serial Port](#)
- [Read Data from Serial Port](#)
- [Send Data to Printer](#)

- [HTTP Request](#)
- [Web Service](#)

File operations group defines the active file related actions:

- [Save Data to File](#)
- [Read Data from File](#)
- [Delete File](#)
- [Browse File/Folder](#)

Flow control group defines various sequences of actions:

- [For Loop](#)
- [For Every Record](#)
- [Try](#)
- [Group](#)

Other group contains specific actions for running the commands, sending custom commands and verifying the licenses:

- [Run Command File](#)
- [Send Custom Commands](#)
- [Verify License](#)

General

Open Label

Open Label specifies the label file that is going to be printed. When the action is executed, the label template opens in memory cache.

The number of concurrently opened labels is not limited. If the label is already loaded and is requested again, Designer determines if a newer version is available, and approves it for printing. The label opens.

Settings group selects the label file.

- **Label name:** label file to be opened. It can be hard-coded – the same label will be printed every time. The option **Variable** enables a variable file name. Select a variable that contains the path and/or label file name.

NOTE Use UNC syntax for network resources. For more information, see the topic [Access to Network Shared Resources](#).

There are four ways to open a label:

- Enter absolute file path.
- Select an existing label from the solution.
- Click **Open** to locate the file on the disk.
- Use a data source to define the file path dynamically.

If the specified label cannot be found, Designer searches for it at alternative locations. For more information on alternative location search, see topic [Search order for the requested files](#).

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Print Label

Print Label executes label printing. The action cannot be used on its own – it must always be nested within the [Open Label](#) action to obtain the reference to the label that is going to be printed.

This allows the user to have multiple labels opened at the same time, so you can specify which label should be printed. When issuing this command, the label prints using the printer driver defined in the label template. If the specified printer driver is not found on the system, the label is printed using the system default printer driver.

TIP: You can override the printer driver using the [Set Printer](#) command.

Designer prints labels in asynchronous mode. This means that as soon as the event pre-processing is complete, and the instructions for the print engine are available, the printing thread takes it over in the background. The control is returned to the event so it can continue executing the next action.

Quantity group defines the number of labels to be printed using the active form.

- **Labels:** sets the number of printed labels.
- **All (unlimited quantity):** labels are printed in different quantities, depending on the design of the label template. Read more about unlimited label printing [here](#).
- **Variable quantity (defined from label variable):** specifies a label variable that defines the label quantity to be printed.

Variable value must be integer. The event doesn't receive the number of labels to be printed so it passes the decision on the quantity to the label template. The label might contain a connection to a database or another data source, which determines the label quantity value.

Advanced group defines label printing details. Click **Show advanced print options** to define the **Advanced** print options:

- **Number of skipped labels:** defines the amount of labels to be skipped on first page. This option is used with [Labels Across](#).
- **Identical label copies:** defines the number of copies for each label in a print job.
- **Label sets:** specifies how many times the entire label printing process should repeat.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Execute Script

Execute Script enables the use of [VBScript](#) or [Python](#) scripts as customizable actions. Use this action if the built-in predefined actions don't meet existing data manipulation requirements.

NOTE Support for VBScript is already available with your Windows system. To install Python support, see [Knowledge Base article KB249](#).

NOTE Select the scripting language in [form properties – additional settings](#).

Script editor offers the following features:

- **Verify syntax:** validates the entered script syntax.
- **Export:** saves the script to a disk. This makes the script reusable.
- **Import:** imports the script from other applications to be used in Designer.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Open Document / Program

Open Document / Program interfaces with external programs and executes them using a command line. External programs execute additional processing and provide the result back to Designer.

This action allows Designer to bind with any third party software that can execute additional data processing or acquire data. External software provides data response by saving it in a file, from where it can be retrieved in variables.

TIP: The user can feed the value of variable(s) to the program by listing them in the command line using square brackets.

File group defines the file to be opened.

- **File name:** location of the file or program to be opened within this action.

Execution Options group sets program opening details.

- **Hide window:** renders the window of the opened program invisible. Because Designer is run as a service application within its own session, it cannot interact with desktop, even if it runs with the privileges of the currently logged user. Microsoft has prevented this interaction in Windows Vista and newer operating systems for security reasons.
- **Wait for completion:** specifies for action execution to wait for this action to be completed before continuing with the next scheduled action.

Enable this option if the action that follows, depends on the result of the external application.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Printer

Set Printer

Set Printer specifies the name of the printer to be used for printing the active label.

NOTE This action overrides the printer defined in the label properties.

This action is useful when printing an identical label on multiple printers. Always nest this action under the [Open Label](#) action to provide the label with the reference on where to change the printer.

This action reads the default settings such as speed and darkness from the selected printer driver and applies them to the label. If you don't use the **Set Printer** action, the label prints on the printer as defined in the label.

WARNING Be careful when changing the printer from one printer brand to another, e.g. from Zebra to SATO, or even from one printer model to another model of the same brand. Printer settings might not be compatible and label printout might not be identical. Also, label design optimizations for the original printer, such as internal counters, and internal fonts, might not be available on the selected printer.

TIP: See section Automatic Font Replacement and its subsection Configuring the Font Mapping to tackle the above mentioned issue.

Printer group specifies the printer name to be used for the current print job.

- **Printer name:** Select it from the list of locally installed printer drivers, or manually enter a printer name. Select **Data source** to dynamically select the printer using a

variable. When enabled, select a variable that contains the printer name when a the action is run.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Define Printer Settings

Define Printer Settings sets properties of the form's selected printer. This way, the user can change printer properties such as printing speed, darkness, and other controls independently from the current printer settings as defined the label, in the printer driver or using the printer hardware settings.

The modifications the user makes using this action are temporary and affect only the current print job. The modifications are not saved in a label or form.

Condition: a scripting Boolean expression. Two results are possible – True or False. Use this option to enable current action only when a term is met. The action is started only if the condition returns True as the result of the expression.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Set Print Job Name

Set Print Job Name specifies the name of the print job file as it appears in the Windows Spooler. A default print job name is the name of the used label file. This action overrides it.

NOTE Always nest the action under the [Open Label](#) action, so it applies to the adequate label file.

Print Job group defines print job name.

- **Name:** sets the print job name. It can be hard-coded, and the same name will be used for each print action. Variable enables a variable file name. Select a variable that contains the path and/or file name when the event happens.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Redirect Printing to File

Redirect Printing to File redirects the print stream to a file. Instead of sending the created print stream to the printer port as defined in the printer driver, the stream is redirected to a file.

You can append the data to an existing file or overwrite it. Capture printer commands to a file using this action.

File group of settings defines how the file selection for redirecting is done.

- **Ask user for file name:** lets the user define the file to print the label to. File browse dialog window opens allowing the user to select the file.
- **Define file name:** predefines the file to print the label to. Each time the action is run, the file is printed to the same file.
 - **File name:** file to which the print stream is redirected to.

NOTE When using this action, make sure your user account has sufficient privileges for accessing the specified folder with read/write permissions.

File write mode group of settings selects how the file is treated in case of repeated redirects.

- **Overwrite the file:** if the specified file already exists on the disk, it is going to be overwritten.
- **Append data to the file:** print stream data is added to the existing data in the provided file.

Persistence controls the continuity of the redirect action. It defines the number of [Print Label](#) actions that are affected by the **Redirect Printing to File** action.

- **Apply to next print action:** specifies for the print redirect to be applicable to the next [Print Label](#) action only (single event).
- **Apply to all subsequent print actions:** specifies for the print redirect to be applicable to all **Print Label** action defined after the current **Redirect Printing to File** action.

NOTE The action only redirects printing. Make sure it is followed by the [Print Label](#) action.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Set Print Parameter

Set Printer Parameter allows fine-tuning the printer driver related parameters. These include parameters such as speed and darkness for label printers, or paper tray for laser printers.

Printer settings are applied to the current printout only and are not remembered during the upcoming event.

TIP: If using the **Set Printer Parameter** action for changing the printer name, make sure you use the **Set Print Parameter** action after it. Before applying the DEVMODE structure to the printer driver, it is obligatory to load the default driver settings first. This is done by the **Set Printer** action. The DEVMODE is only compatible with the DEVMODE of the same printer driver.

Print Parameters group allows fine tuning before printing.

- **Paper bin:** name of the paper bin that contains the label media. This option is usually used with laser and ink jet printers with multiple paper bins. The provided name of the paper bin must match the name of the bin in the printer driver. Check the printer driver properties for more details.
- **Print speed:** defines printing speed. This setting overrides the setting defined with label. The provided value must be in the range of accepted values.
- **Darkness:** defines the darkness of the printed objects on the paper and overrides setting from the label. The provided value must be in range of accepted values.
- **Print offset X:** applies horizontal offset. The label printout will be repositioned by the specified number of dots in the horizontal direction. Negative offset can be defined.
- **Print offset Y:** applies vertical offset. The label printout will be repositioned by the specified number of dots in the vertical direction. Negative offset can be defined.

Advanced group customizes the printer settings sent with the print job.

Printer settings, such as printing speed, darkness, media type, offsets and similar, can be defined as follows:

- Defined in the label.
- Recalled from the printer driver.
- Recalled from the printer at print time.

The supported methods depend on the printer driver and printer capabilities. The printing mode (recall settings from label or driver or printer) is configurable in the label design. You might need to apply these printer settings at print time – they can vary with each printout.

EXAMPLE A single label should be printed using a variety of printers, but each printer requires slightly different parameters. The printers from different manufacturers don't use the same values to set the printing speed or temperature. Additionally, some printers require vertical or horizontal offset to print the label to the correct position. During the testing phase, you can determine the optimal settings for every printer you intend to use and apply them to a single label template just before printing. This action will apply the corresponding settings to each defined printer.

This action expects to receive the printer settings in a DEVMODE structure. This is a Windows standard data structure with information about initialization and environment of a printer.

Printer settings option applies custom printer settings. The following inputs are available:

- **Fixed-data Base64-encoded DEVMODE.** In this case, provide the printer's DEVMODE encoded in Base64-encoded string directly into the edit field. When executed, the action will convert the Base64-encoded data back into the binary form.
- **Variable-data Base64-encoded DEVMODE.** In this case, the selected variable must contain the Base64-encoded DEVMODE. Enable Variable and select the appropriate variable from the list. When executed, the action will convert the Base64-encoded data back into the binary form.
- **Variable-data binary DEVMODE.** In this case, the selected variable must contain the DEVMODE in its native binary form. Enable Variable and select the appropriate variable from the list. When executed, the action will use the DEVMODE as-is, without any conversion.

NOTE If the variable does not provide binary DEVMODE, make sure that the selected variable is defined as binary variable in the configuration.

NOTE Make sure the [Set Printer](#) action is defined in front of this action.

Extracting the DEVMODE structure

DEVMODE structure can be extracted from the registry.

To help you test and use the [Set Printer Parameter](#) action, the application has been provided that retrieves the DEVMODE of the selected printer and save it to file or Base64-encode it for you. You can find the application `GetPrinterSettings.exe` on the NiceLabel 2017 DVD and at NiceLabel website.

Using the application interactively

Run the application, select the printer for which you need a DEVMODE structure and click the **Get Printer Settings** button. The DEVMODE will be provided as Base64-encoded string. You can paste it into the [Set Printer Parameter](#) action.

Using the application with command-line parameters

In this case you can control the application with the command-line parameters. **Syntax:**

```
GetPrinterSettings.exe <printer_name> <file_name> [base64]
```

- `printer_name`: name of the printer driver as available in the Windows system.
- `file_name`: name of the file that will contain the extracted DEVMODE
- `base64`: optional parameter. If provided, the DEVMODE will be encoded into Base64 string, otherwise the DEVMODE will be provided as the binary data

For example:

Save DEVMODE for printer "Avery AP 5.4 300DPI" as binary data in file "devmode1".

```
GetPrinterSettings.exe "Avery AP 5.4 300DPI" c:\temp\devmode1
```

Save DEVMODE for printer "Avery AP 5.4 300DPI" as Base64-encoded data in file "devmode2".

```
GetPrinterSettings.exe "Avery AP 5.4 300DPI" c:\temp\devmode2 base64
```

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled**: specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition**: action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure**: specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable**: enables the user to define the **Data Source** (variable) to save the error to.

Redirect Printing to PDF

Redirect Printing to PDF diverts the print stream to a PDF file. Instead of sending the label to a printer, the printout is redirected to a PDF file.

The print stream data can be appended to the existing file, or it may overwrite an existing file. The PDF document retains the exact label dimensions as defined during label designing. The rendering quality of graphics in the PDF matches the resolution of the target printer and desired printout size.

The action will redirect printing only, so make sure it is followed by the [Print Label](#) action.

File group defines the redirect file.

- **File name:** specifies the file name for diverting the print job to. If hard-coded, the printing is redirected to the specified file every time.
- **Overwrite the file:** if the specified file already exists on the disk, it is going to be overwritten (selected by default).
- **Append data to the file:** The job file is appended to the existing data in the provided file (deselected by default).

Persistence allows controlling the persistence of the redirect action. Define the number of [Print Label](#) actions that are affected by the **Redirect Printing to File** action.

- **Apply to next print action:** specifies for the print redirect to be applicable to the next [Print Label](#) action only (single event).
- **Apply to all subsequent print actions:** specifies for the print redirect to be applicable to all **Print Label** action defined after the current **Redirect Printing to File** action.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Printer Status

Printer Status communicates with the printer to acquire its real-time state, and contacts the Windows Spooler for additional information about the printer and its jobs.

As a result the information about errors, spooler status, number of jobs in the spooler is collected. This uncovers the potential errors.

The following prerequisites must be met to enable the live printer status:

- Use a NiceLabel Printer Driver to receive detailed printer status information. With other printer drivers, only the information retrieved from the Windows Spooler can be seen (including live printer status).
- The printer must be capable of reporting the live status. Printer models supporting bidirectional communication are listed on the NiceLabel Download web page.
- The printer must be connected using an interface that supports bidirectional communication.
- The bidirectional support must be enabled in the **Control Panel > Hardware and Sound > Devices and Printers > driver > Printer Properties > Ports tab > Enable bidirectional support**.
- If using network-connected label printer, make sure the Advanced TCP/IP Port (not Standard TCP/IP Port!) is used. For more information, see [Knowledge Base article KB189](#).

Printer group selects the printer.

- **Printer name** specifies the printer name to be used for the current print job.

Data Mapping group sets the parameters that are returned as a result of the **Printer Status** action.

WARNING The majority of the below listed parameters are supported only when using NiceLabel Printer Drivers. With other printer drivers, only spooler-related parameters are available.

- **Printer status:** specifies the printer live status as a string. If the printer reports multiple states, all states are merged into a single string, delimited by comma ",". If there are no reported printer issues, this field is empty. Printer status might be set to **Offline**, **Out of labels** or **Ribbon near end**. Since there is no standardized reporting protocol, each printer vendor uses different status messages.
- **Printer error:** specifies the boolean (true/false) value of the printer error status.
- **Printer offline:** specifies the boolean (true/false) value of the printer offline status.
- **Driver paused:** specifies the boolean (true/false) value of the driver pause status.
- **NiceDriver driver:** specifies the boolean (true/false) value of the NiceLabel Printer Driver status. Provides information whether or not the selected driver is a NiceLabel Printer Driver.
- **Spooler status:** specifies the spooler status in the form of a string as reported by the Windows system. The spooler can simultaneously report several statuses. In this case, the statuses are merged using comma ",".
- **Spooler status ID:** specifies the spooler status in the form of a number, as is reported by the Windows system. The spooler can simultaneously report several statuses. In this case, the returned status IDs contain all IDs as flags. For example, value 5 represents status IDs 4 and 1, which translates to "**Printer is in error**,

Printer is paused". Refer to the [Spooler Status ID](#) table. The action returns a decimal value. Since the values in the table below are in hex, a conversion is necessary before parsing the response.

- **Number of jobs in the spooler:** specifies the number of jobs that are currently in the spooler for the selected printer.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Store Label to Printer

Store Label to Printer saves a label template in the printer memory. The action is a vital part of [Store/Recall printing mode](#).

Advanced options for storing label to printer group defines the printer name and store variant.

- **Label name to be used on the printer:** defines the name to be used for storing the label template in the printer memory.
- **Store variant:** specifies how the label templates should be stored. Enter the location manually. If no variant is defined, the first available variant is used. To select between the available options, use the **Store variant** drop down list under **Label Properties -> Printer** tab.

WARNING When storing the label to a printer, it is recommended to leave the label name under the advanced options empty. This prevents label name conflicts during the recall label process.

NOTE To make sure the stored label samples are not lost after power cycling the printer, store them at non-volatile locations.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Form

Open Another Form

Open Another Form action opens another from the same solution or a form from a disk.

Settings group includes the following options:

- **Navigate back to previously opened form:** reopens the preceding form when the **Open Another Form** action is run.
- **Open form:** defines a form to be opened when the **Open Another Form** action is run.

There are four ways to open a form:

- Enter the absolute file path.
- Select an existing form from the solution.
- Click **Open** to locate the file on the disk.
- Use a data source to define the file path dynamically.

Form data sources group allows resetting the data source values.

- **Reset data source values:** resets the **Data Source** values after each execution of the **Open Another Form** action.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Message

Message opens a message window containing a custom message.

Content group defines caption and the message content.

- **Caption:** specifies the window title.
- **Message:** specifies the custom message content.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Quit

Quit closes the form.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Move Focus

Move Focus moves the focus to a specified object.

Settings group defines focus movement:

- **Move focus to first object in tab order:** sets focus on the first object in the defined order after running the form.
- **Move focus to selected object** places focus after running the form on the selected object.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Set Object Property

Set Object Property sets properties of form object, like width, height and color.

Settings group defines the properties to be set:

- **Object name:** form object to be edited. Drop-down list contains objects on the form.
- **Property:** defines the property to be set. The set of available properties depends on the currently selected object

Descriptions of available properties are available in [topics about the form objects](#).

- **Value** defines the property value.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Translate Form

Translate Forms translates all strings on a form to the selected language.

Translate form settings group selects the language and creates the translation file.

- **Language:** language to be used on the translated form. The language name is defined in the first row of the translation file.

The language name in the translation file is user-configurable. Use the same ID (name) in the action as you have defined in the translation file. The language name can be fixed or variable. Its usage depends on the language selection type you that is used on the form.

- **Fixed name:** hard-coded language name which must match the name in the first row of the translation file.
- **Variable name:** Example is a drop-down box with language names. When the user changes the language in the list, the onChange event executes the Translate Form action. The drop-down box saves the user selection in a variable, which is used for the action.

- **Translation file:** file that contains source strings and translations into various languages. This is a structured text file, similar to a CSV file.
- **Create translation file:** click this button to create the translation file containing the source and translated strings.

Translation File Structure is a text file with UTF-8 encoded data. It is similar to comma-separated-values (CSV).

Formatting Rules are mandatory. Always follow the below listed rules.

- The first line contains the language ID.
 - The first field is always named **Source**. Do not change it.
- The names of other fields in the first row are user-configurable. Use the suggested names, such as "Language 2" and "Language 3", or replace them with whatever describes the language better, such as "German", "French", "Chinese", etc.
- All lines that follow the first line are lines with the translations from the original language. The first field contains the original string, the next fields in the same line contain translation to other languages. The first line specifies in which order the translation should follow the source string.
- All values are enclosed with double-quote characters (").
- All values are delimited by a semicolon character (;).

- If you have multiline text objects in the form, the newline (<CR><LF>) will be encoded as special string \$NEWLINE\$.
- If you leave the translation empty, the Source string is used.

EXAMPLE OF TRANSLATION FILE:

```
Source"; "DE"

"&Print"; "&Druck"

"Customize$NEWLINE$your$NEWLINE$printing$NEWLINE$forms"; "
Anpassen$NEWLINE$Sie$NEWLINE$Ihre$NEWLINE$Druckformen"

"Printer: "; "Drucker"

"Quantity"; "Menge"

"SAMPLE"; "PROBE"

"Se&ttings"; "Einstellungen"

"Translate"; "Übersetzen"

"www.nicelabel.com/solutions"; ""
```

Translating Strings

When using the Translate Form action anywhere in your form, all strings of the form are automatically saved to the translation file whenever you save the form. This ensures that the translation file is always up to date with your form.

The translation file is Unicode-aware text file. You can edit it in any text editor, but you might have troubles recognizing the fields, because their values are semicolon-separated and not aligned one below another.

You can also open the file in a spreadsheet application, such as Microsoft Excel. In this case, the fields belonging to particular language are displayed in the same column of data, and much easier to edit.

NOTE Spreadsheet applications might change the input file structure of the translation file. In this case, you will have to reformat the data yourself after you saving the translation file.

EXAMPLE Microsoft Excel will save the translation file as CSV. The fields lose the double quotes around the values and will be delimited by comma (,) instead of semicolon (;). You will have to convert commas into semicolons and put double quotes around fields. This can be done with a few search & replace actions.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Variables

Set Variable

Set Variable assigns a new value to the selected variable when the form is run.

Variable group includes the following options.

- **Name:** name of variable that should have the value changed.
- **Value:** value to be set to a variable.

The allowed content types are fixed content, mix of fixed and variable content, or variable content alone. Detailed description is available in topic [Combining Values in an Object](#).

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the

same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Save Variable Data

Save Variable Data stores values of a single or multiple variables in an associated file.

TIP: To recall the saved data, use action [Load Variable Data](#). The values are saved using CSV format with the first line containing variable names. If the variables contain multiline values, the new line characters (CR/LF) are encoded as `\n\r`.

Settings group defines the file name.

- **File name:** file to save the variable data to. If the name is hard-coded, values are saved into the same file each time.

If file exists group offers additional options:

- **Overwrite the file:** overwrites the existing data with new variable data. The old content is lost.
- **Append data to the file:** appends the variable values to the existing data files.

File Structure group defines the CSV file parameters:

- **Delimiter:** specifies delimiter type (tab, semicolon, comma or custom character). Delimiter is a character that separates the values.
- **Text qualifier:** specifies the character that qualifies content as text.
- **File encoding:** specifies character encoding type to be used in the data file. **Auto** defines the encoding automatically. If required, the preferred encoding type can be selected from the drop down list.
- **Add names of variable in the first row:** places the variable name in the first row of the file.

Variables group defines the variables whose value should be saved.

- **All variables:** variable data of all variables is saved.
- **Selected variables:** variable data of listed variables is saved.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Load Variable Data

Load Variable Data loads values of a single or multiple variables from the associated file as they were saved using the [Save Variable Data](#) action.

Settings group defines the file name.

- **File name:** specifies the file for the variable data to be loaded from. If the name is hard-coded, the values are loaded from the same file each time.

File Structure group settings must reflect the structure of the saved file from the [Save Variable Data](#) action.

- **Delimiter:** specifies delimiter type (tab, semicolon, comma or custom character). Delimiter is a character that separates the values.
- **Text qualifier:** specifies the character that qualifies content as text.
- **File encoding:** specifies the character encoding type used in the data file. **Auto** defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

Variables group defines the variables whose value should be loaded.

- **All variables:** specifies all defined variables in the data file to be read.
- **Selected variables:** specifies the specific selection of variables to be read from the data file.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

String Manipulation

String Manipulation defines the value format of the selected variables. this action's purpose is to fine-tune the data values.

Variables group defines the variables to which the string manipulation applies.

- **All variables:** specifies all the defined variables in a data file to be formatted.
- **Selected variables:** specifies a selection of variables to be formatted from the data file.

Format Text group defines string manipulation functions that apply to the selected variables or fields. Multiple functions can be used. The functions apply in the same order as seen in the editor – from top to bottom.

- **Delete spaces at the beginning:** deletes all space characters (decimal ASCII code 32) from the beginning of the string.
- **Delete spaces at the end:** deletes all space characters (decimal ASCII value 32) from the end of a string.
- **Delete opening closing characters:** deletes the first occurrence of the selected opening and closing characters that are found in the string.

EXAMPLE: When using "{" as opening character and "}" as closing character, the input string {{selection}} is converted to {selection}.

- **Search and replace:** executes standard search and replace function upon the provided values for *find what* and *replace with*. Regular expressions are supported.

NOTE There are several implementations of the regular expressions in use. Designer uses .NET Framework syntax for the regular expressions. For more information, see [Knowledge Base article KB250](#).

- **Replace non printable characters with space:** replaces all control characters in the string with space character (decimal ASCII code 32). Non printable characters are characters with decimal ASCII values between 0–31 and 127–159.
- **Delete non printable characters:** deletes all control characters in the string. The non-printable characters are characters with decimal ASCII values between 0–31 and 127–159.
- **Decode special characters:** decodes the characters (or control codes) that are not available on the keyboard, such as Carriage Return or Line Feed. Designer

uses a notation to encode such characters in human-readable form, such as <CR> for Carriage Return and <LF> for Line Feed. This option converts special characters from NiceLabel syntax into actual binary characters.

EXAMPLE: When you receive the data "<CR><LF>", Designer uses it as plain string of 8 characters. You will have to enable this option to interpret and use the received data as two binary characters CR (Carriage Return – ASCII code 13) and LF (Line Feed – ASCII code 10).

- **Search and delete everything before:** finds the provided string and deletes all characters in front of the defined string. The string can also be deleted.
- **Search and delete everything after:** finds the provided string and deletes all characters behind the defined string. The string can also be deleted.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Data And Connectivity

Execute SQL Statement

Execute SQL Statement sends SQL commands to an SQL server and collects the results.

TIP: Use commands SELECT, INSERT, UPDATE, and DELETE.

NOTE To use a variable value within an SQL statement, enter colon (:) in front of its name. This gives Designer a signal that a variable name follows.

Database Connection defines the database connection that will be used for the statement.

SQL Statement group defines an SQL statement or query to be executed.

TIP: Statements from Data Manipulation Language (DML) are allowed to execute queries upon existing database tables. Use standard SQL statements, such as SELECT, INSERT, DELETE and UPDATE, including joins, function and keywords. The statements DDL language to create databases and tables (CREATE DATABASE, CREATE TABLE), or to delete them (DROP TABLE) are not permitted.

- **Test:** opens the **Data Preview** section. Simulate execution (selected by default) tests the execution of SQL statements. Click **Execute** to run the simulation.
- **Insert variable:** inserts the predefined variables into the SQL statement.
- **Export/Import:** enables exporting and importing SQL statements to/from an external file.
- **Execution mode:** specifies the explicit mode of execution.
 - **Automatic:** the application determines the action automatically.
 - **Returns set of records (SELECT):** receive the data set with records.
 - **Does not return set of records (INSERT, DELETE, UPDATE):** You are executing a query that does not return the records. Either insert new records, delete or update the existing records. The result is a status response reporting the number of rows were affected by your query.

With certain complex SQL queries, it becomes increasingly difficult to automatically determine what the supposed action is. If the built-in logic encounters troubles identifying the action's intent, select the main action manually.

Save result to variable defines the variable to store the SQL statement result.

Retry on failure allows you to configure the action to continually retry establishing the connection to the database server in case the first attempt was not successful. If the action will fail to connect in all defined number of attempts, the error is raised.

- **Retry attempts:** specifies the number of tries to connect to the database server.
- **Retry interval:** specifies the time period to wait before trying to reconnect to the database.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Refresh Table

Refresh Table rereads a database table.

Table group selects the database table to be reread.

- **Table:** defines an existing table to be reread or creates a new one using the Step-by-Step Database Wizard.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Import Data into Table

Import Data into Table action reads data from a formatted CSV text file and imports it into an SQL database.

NOTE Before using this action, a connection to the SQL database must already be set. The action does not work with file-based databases, such as Microsoft Access, or data files like Microsoft Excel, or plain text files. Use a server-based SQL database, such as Microsoft SQL Server.

The following rules apply to this action:

- The table must already exist within the SQL database.
- The table must contain PRIMARY KEY.
- The first line in a text file must define field names.
- The field names in the text file must match field names in the database table.
- If the text file does not provide a value for some field, NULL is written to the database. If the field does not accept NULL values, an empty string ("") is written.
- Setting values for auto-incremental fields are ignored. The database provides value for such field.
- If the value from text file does not match the structure of the field, the action is canceled and an error message is displayed. For example, when trying to enter alpha-numerical value into numerical field.
- If you filter records on the form and display only records matching certain condition, you can only import records that either do not provide value for the filter field, or provide the same value for the filter as defined with the form.
- Only filters with condition "equal" , not "greater than", "less than", "contains" or similar are permitted.
- If the text file contains fields not defined in the SQL database, the import will ignore them. Only known fields will be imported.

Settings group selects the table.

- **Table** defines a predefined table from the drop down menu or creates a new one using the Step-by-Step Database Wizard.

File Text Structure group specifies the text database parameters:

- **Delimiter:** specifies the delimiter type in the data file. Select a predefined delimiter, or create a custom one.
- **Text qualifier:** specifies the text qualifier. Select a predefined delimiter or insert a custom one.
- **File encoding:** specifies the character encoding type used in the data file. **Auto** defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Send Data to TCP/IP Port

Send Data to TCP/IP Port sends the data to any external device that supports TCP/IP connection on a predefined port number.

The action establishes connection with a device, sends the data and terminates the connection. Connection and communication are governed by the handshake that occurs between a client and server while initiating or terminating a TCP connection.

Connection Settings group sets connection details:

- **Destination (IP address:port):** destination address and port of the TCP/IP server. Hard-code the connection parameters and use fixed host name or IP address or use variable connection parameters by clicking the right arrow and selecting a predefined variable. For more information, see topic [Combining Values in an Object](#).
- **Disconnect delay:** prolongs the connection with the target socket for the defined time intervals after the data has been delivered. Certain devices require more time to process the data. Insert the delay value manually or click the arrows to increase or decrease it.

Content defines the content to be sent to the TCP/IP server.

TIP: Use fixed content, mix of fixed and variable content, or variable content alone. To enter variable content, click the button with arrow to the right of data area and insert a variable from the list. For more information, see the topic [Combining Values in an Object](#).

- **Data:** content to be sent outbound.
- **Encoding:** encoding type for the sent data. **Auto** defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Send Data to Serial Port

Send Data to Serial Port sends data to a serial port.

Use this action to communicate with external serial-port devices. Make sure the port settings on both ends match. Serial port can be used by a single running application. To successfully use the port from this action, no other application should use the port, not even a printer driver.

Port group defines the serial port.

- **Port name:** name of the port to which an external device connects to. This can be a hardware COM port or a virtual COM port.

Port Settings group defines additional port connection settings:

- **Bits per second:** speed rate used by the an external device to communicate with the PC. The usual alias used with the setting is "baud rate". Select the value from the drop down menu.
- **Data bits:** number of data bits in each character. 8 data bits are almost universally used in newer devices. Select the value from the drop down menu.
- **Parity:** method of detecting errors in a transmission. The most common parity setting, is "none", with error detection handled by a communication protocol (flow control). Select the value from the drop down menu.
- **Stop bits:** halts the bits sent at the end of every character allowing the receiving signal hardware to detect the end of a character and to resynchronize with the character stream. Electronic devices usually use a single stop bit. Select the value from the drop down menu.
- **Flow control:** serial port may use interface signals to pause and resume the data transmission.

EXAMPLE A slow device might need to handshake with the serial port to indicate that data should be paused while the device processes received data.

Content group defines the content to be sent to serial port.

TIP: Fixed content, mix of fixed and variable content, or variable content alone are permitted. To enter variable content, click the button with arrow to the right of data area and insert a variable from the list. For more information, see the topic [Combining Values in an Object](#).

- **Data:** content to be sent outbound.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Read Data From Serial Port

Read Data From Serial Port collects the data received on the serial port (RS-232) and saves it to a selected variable. This action communicates with external serial port devices.

Port group selects the serial port.

- **Port name:** port to which the external device is connected. The possibilities are hardware COM port or virtual COM port.

Port Settings group defines additional port connection settings:

- **Bits per second:** speed rate used by the an external device to communicate with the PC. The usual alias used with the setting is "baud rate".
- **Data bits:** specifies the number of data bits in each character. 8 data bits are almost universally used in newer devices.
- **Parity:** specifies the method of detecting errors in a transmission. The most common parity setting, is "none", with error detection handled by a communication protocol (flow control).

- **Stop bits:** halts the bits sent at the end of every character allowing the receiving signal hardware to detect the end of a character and to resynchronize with the character stream. Electronic devices usually use a single stop bit.
- **Flow control:** serial port may use interface signals to pause and resume the data transmission.

EXAMPLE A slow device might need to handshake with the serial port to indicate that data should be paused while the device processes received data.

Options group includes the following settings:

- **Read delay:** optional delay when reading data from serial port. After the delay, the entire content of the serial port buffer is read. Enter the delay manually or click the arrows to increase or decrease the value.
- **Send initialization data:** specifies the string that is sent to the selected serial port before the data is read. This option enables the action to initialize the device to be able to provide the data. The option can also be used for sending a specific question to the device, and to receive a specific answer. Click the arrow button to enter [special characters](#).
- **Data Extraction:** extracts the defined parts of the received data.
 - **Start position:** starting position of extracted data is defined by entering a number.
 - **End position:** ending position of extracted is defined by entering a number.

Result group defines a variable for data storing.

- **Save data to variable:** variable selection for storing the received data.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the

same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Send Data to Printer

Send Data to Printer sends data to a selected printer.

The action is useful for sending pre-generated printer streams to any available printer. Designer uses the installed printer driver in pass-through mode in order to be able to send data to the target LPT, COM, TCP/IP or USB port, to which the printer is connected.

Printer group selects the printer.

- **Printer name:** name of the printer to send the data to. Select the printer from the drop down list of locally installed printer drivers, or enter a custom printer.

Data Source defines the content to be sent to printer.

- **File name:** path and file name of the file containing a printer stream. Contents of the specified file is sent to a printer. Select **Data source** to define the file name dynamically using a variable value.
- **Variable:** variable that contains the printer stream.
- **Custom:** defines custom content to be sent to a printer. Fixed content, mix of fixed and variable content, or variable content alone are permitted. To enter variable content, click the button with arrow to the right of data area and insert a variable from the list. For more information, see the topic [Combining Values in an Object](#).

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

HTTP Request

HTTP Request sends data to the destination Web server using the selected HTTP method. HTTP and HTTPS URI schemes are permitted.

HTTP functions works as a request–response protocol in the client–server computing model. With this action Designer takes a role of the client, communicating with the remote server. This action submits the selected HTTP request message to the server. The server returns a response message containing a completion status information about the request and may also contain the requested content in the message body.

Connection Settings group sets the connection parameters.

NOTE This action supports Internet Protocol version 6 (IPv6).

- **Destination:** address, port and destination (path) of the Web server.

If the Web server runs on default port 80, skip the port number. Hard-code the connection parameters and use a fixed host name or IP address. Use a variable value to define this option dynamically. For more information, see topic [Combining Values in an Object](#).

- **Request method:** available request methods.
- **Timeout:** timeout (in ms) in which the connection to the server should be established.
- **Save status reply in a variable:** variable to store the status code received from the server.
- **Save data reply in a variable:** variable to store the data received from the server.

Content define the contents to be sent to a Web server.

- **Data:** content to be sent outbound. Fixed content, mix of fixed and variable content, or variable content alone are permitted. To enter variable content, click the button with arrow to the right of data area and insert variable from the list. For more information, see topic [Combining Values in an Object](#).
- **Encoding:** encoding type for the sent data.

Auto defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

- **Type:** Content-Type property of the HTTP message. If no type is selected, the default application/x-www-form-urlencoded type is used. If an appropriate type is not listed, define a custom one.

Additional HTTP Headers are requested by certain HTTP servers (especially for REST services).

- **Additional headers:** hard coded headers or headers obtained from variable values. To access the variables, click the small arrow button to the right of the text area. For more information, see the topic [Combining Values in an Object](#).

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Web Service

Web Service connects to a SOAP server and executes the methods on it. This action sends inbound values to the Web service and collects the results.

Web Service Definition group includes the following settings:

NOTE This action supports Internet Protocol version 6 (IPv6).

- **WSDL:** location of Web Service Description Language (WSDL) definition. This is XML-based interface description language that describes the functionality offered by the Web service. The WSDL is usually provided by the Web service itself. Typically you would enter the link to WSDL and click the **Import** button to read the definition. If you have troubles getting WSDL from the online resource, save the WSDL to file and enter the path with file name to load methods from it. Designer automatically detects if the remote Web Service uses document or RPC syntax and communicates appropriately.
- **Address:** address where the Web Service is published. Initially, this information is retrieved from the WSDL, but can be updated it before the action is executed. This is helpful for development / test / production environments, where the same list of actions is used, but with different names of servers where Web Services run. Fixed content, mix of fixed and variable content, or variable content alone are permitted. To enter variable content, click the button with arrow to the right of data area and insert variable from the list. For more information, see the topic [Combining Values in an Object](#).

- **Method:** methods (functions) which are available in a selected Web service. The list is automatically populated by the WSDL definition.
- **Parameters:** input and output variables for the selected method (function). The inbound parameters expect an input. For testing and troubleshooting reasons you can enter a fixed value and see the preview result on-screen. But typically you would select a variable for inbound parameter. Value of that variable will be used as input parameter. The outbound parameter provides the result from the function. You must select the variable that will store the result.
- **Timeout:** timeout after which the connection to a server is established.

Authentication enable basic user authentication. This option defines the user credentials that are necessary to establish an outbound call to a remote web service.

- **Enable basic authentication:** enables defining the Username and Password that can be entered manually or defined by variable values. Select **Data sources** to define the variables.
- **Show password:** uncovers the masked Username and Password characters.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

File Operations

Save Data to File

Save Data to File saves data streams or a variable value in an external file. Make sure the application has write permission for the given folder.

File group sets the file related details.

- **File name:** name of the file that stores the data. **File name** can be hard-coded – the data is stored in the same file with each action. **Data source** dynamically defines the **File name**.

If file exists group handles options in case of existing file.

- **Overwrite the file:** overwrites existing data with new data. The old content is lost.
- **Append data to the file:** appends the variable values to the existing data files.

Content group defines the content as provided in the text area. Fixed values, variable values and special characters are permitted. To enter variables and special characters, click the arrow button to the right of the text area. For more information, see the topic [Combining Values in an Object](#).

- **Encoding:** encoding type for the sent data. **Auto** defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Read Data from File

Read Data from File reads the content of a specified file and saves it in a variable. This variable can be later used as dynamic content for another object.

File group sets the file related details.

- **File name:** name of the file to read the data from. **File name** can be hard-coded – the data is read from the same file with each action. **Data source** dynamically defines the **File name**.

Content group sets the file content related details.

- **Variable:** variable that stores the file content. At least one variable should be defined.
- **Encoding:** encoding type for the sent data. **Auto** defines the encoding automatically. If needed, select the preferred encoding type from the drop down list.

Retry on Failure group defines how the action should continue if the specified file becomes inaccessible.

TIP: Designer might not be able to access the file, because it is locked by another application. If an application still writes data to the selected file and keeps it locked in exclusive mode, no other application can open it at the same time, not even for reading. Other possible causes for action retries: file doesn't exist (yet), folder does not exist (yet), or the service user doesn't have privileges to access the file.

- **Retry attempts:** defines the number of retry attempts for accessing the file. If the value is set to 0, no retries will be made.
- **Retry interval:** time interval between individual retries in milliseconds.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Delete File

Delete File deletes file from disk.

File group sets the file related details.

- **File name:** the name of the file to be deleted. **File name** can be hard-coded. **Data source** dynamically defines the **File name**.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Browse File/Folder

Browse File/Folder opens the system browse for file or folder dialog.

Dialog group sets the browsing preferences.

- **Browse for:** selects between browsing for a file or folder.
- **Filter:** file type to be located. Enter the file type manually, define the filters using a **Define File Filters** dialog or select **Data source** to determine the filter dynamically using a variable value. The **Define File Filters** dialog allows the user to:
 - **List the filters.** Each filter is identified with a **Filter Name** and **Filter** type.
 - **Manage the existing filters** using **Add**, **Delete**, **Move up** and **Move down** buttons.
- **Initial directory:** directory to be opened.
- **Dialog title:** title of the file browser window.

Output data source group selects a variable for file/folder path storing.

- **Save path to:** existing or new variable for the file/folder path to be saved to.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Flow Control

For Loop

For Loop executes subordinate nested actions for multiple times. All of the nested actions are executed in a loop for as many times as defined by the difference between the start and the end value.

Loop Settings group includes the following options:

- **Start value:** loop starting point reference. A negative value is permitted. Select **Data source** to define the start value dynamically using a variable value. Select a variable containing a numeric value for start.
- **End value:** ending point reference. A negative value is permitted. Select **Data source** to define the start value dynamically using a variable value. Select a variable containing a numeric value for start.
- **Save loop value to a variable:** saves the current loop step value in an existing or a new variable. The loop step value is allowed to contain any value between start and end value. Save the value in order to reuse it in another action to identify the current iteration.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

For Every Record

For every record executes subordinate nested actions multiple times. All of the nested actions are executed in a loop for as many records as present in the form table.

Settings group selects the records.

- **Form table:** form table that contains records for which an action should repeat.
- **Use all records:** repeats an action for all records in a defined table.
- **Use selected record:** repeats an action for the selected records only.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Try

Try allows easy monitoring of errors while the actions are being executed. If an error occurs, a different set of actions is run.

The action creates **Do and On error** placeholders for actions. All actions that should be executed, must be placed inside the **Do** placeholder. If no errors are detected when executing actions from **Do** placeholder, these are the only actions that are executed.

However, if an error does occur, the execution of actions from **Do** placeholder stops and the execution switches over to actions from **On error** placeholder.

EXAMPLE

If any of the actions in the Do placeholder fail, the action execution stop and resumes with the actions in the On Error placeholder.

If Try would be placed on its own, it would terminate the execution. In this case, Try is nested under the For loop action. Normally, any error in Do placeholder would also stop executing the For loop action, even if there are still further steps until For loop should complete. In this case the Save Data to File would also not execute. By default, any error breaks the entire action processing.

However, you can also continue with the execution of the next iteration in For loop action. For this to happen, you have to enable Ignore failure in the Try action. If the data from current step in For Loop causes an error in Do placeholder, actions from On Error execute. After that, the Save Data to File in level 2 execute and then the For loop action continues to execute for the next iteration.

TIP: This action provides easy error detection and execution of "feedback" or "reporting" actions. For example, if an error happens during processing, the action enables you to send out a warning.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Group

Group combines multiple actions in a logical group. It enables you to define a condition for an entire set of actions instead of defining conditions individually for each separate action. Group action also helps you logically separate the actions, add descriptions, and grouped actions such as copy, paste, etc.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Other

Run Command File

Run Command File executes the commands in the selected command file. All types of files provide commands that Designer executes in order from top to bottom. Command files usually provide data for a single label – files of any complexity level can be defined.

File type specifies the type of the command file to be executed.

File name: name of command file to be run. **File name** can be hard-coded – the data is read from the same file each time. Select **Data source** dynamically defines the **File name**.

TIP: Use UNC syntax for network resources. For more information, see topic [Access to Network Shared Resources](#).

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Send Custom Commands

Send Custom Commands executes the entered custom commands.

Always nest this action under the [Open Label](#) action. It provides reference to the label to which the commands apply.

NOTE The majority of custom commands is available through individual actions. As a result, custom commands are not required.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Verify License

Verify License reads the activated license and executes the actions nested below this action only if a certain license type is used.

TIP: This action protects the current configuration from being run on unauthorized computers.

The license key that activates the software can also encode a Solution ID. This is a unique number that identifies the solution provider who sold the NiceLabel Designer license. If the configured Solution ID matches the Solution ID encoded in the license, the target computer is allowed to run the nested actions, effectively limiting execution to licenses sold by the solution provider.

Solution ID defines the ID number of the licenses that are allowed to run the nested actions.

- If the entered value is not the Solution ID that is encoded in the license, the nested actions is not executed.
- If the entered value is set to 0, the actions execute if any valid license is found.

Action Execution and Error Handling

Each action in **Actions Editor** can be set as a conditional action. Conditional actions only run when the defined conditions allow them to be run. To define these conditions, click **Show execution and error handling options**.

Execution options are:

- **Enabled:** specifies if the action is enabled or disabled. Only enabled actions are executed.

This functionality may be used while testing a form.

- **Condition:** action execution is defined by a single-line programming expression that provides a Boolean value (`true` or `false`). If the result of the expression is `true`, the action executes.

Condition offers a way to avoid executing actions every time.

Error handling options are:

- **Ignore failure:** specifies whether an error should be ignored or not. With **Ignore failure** enabled, the execution of actions continues even if the current action fails.

NOTE Nested actions that depend on the current action do not execute in case of a failure. The execution of actions continues with the next action on the same level as the current action. The error is logged, but does not break the execution of the action.

- **Save error to variable:** enables the user to define the **Data Source** (variable) to save the error to.

Combining Values In An Object

Certain objects accept multiple values as their content. Such content can be a combination of fixed values, variables and special characters (control codes). The objects that accept combined values are identified by a small right arrow button on the right side of the object. Click the arrow button to enter either a variable or a special character.

- **Using fixed values.** Enter a fixed value for the variable.

This is a fixed value.

- **Using fixed values and data from variables.** Combined values may contain variables and fixed values. The variable names must be enclosed in square brackets `[]`. Enter the variables manually or insert them by clicking the arrow button to the right. During the processing time, the values of variables are merged together with fixed data and used as the object content. In the example below, the content is merged from three variables and fixed data

items.

```
[variable1] // This is fixed value [variable2][variable3]
```

- **Using special characters.** Special characters are supported with combined values. You can enter the special characters manually, or insert them using a drop down list. In this case, the value of `variable1` is merged with fixed data and form-feed binary character. The list of available special characters is available [here](#).

```
[variable1] Form feed will follow this fixed text <FF>
```

Access To Shared Network Resources

This topic describes best practice steps to use shared network resources.

User Privileges For Service Mode

The execution component of Designer runs in service mode under specified user account inheriting access privileges of that account.

To be able to open label files and to use printer drivers in Designer, the associated user account must be granted sufficient privileges.

UNC Notation For Network Shares

When accessing the file on a network drive, use the UNC syntax and not the mapped drive letters. UNC is a naming convention to specify and map network drives. Designer will try to replace the drive-letter syntax with the UNC syntax automatically.

EXAMPLE If the file is accessible as `G:\Labels\label.lbl`, refer to it in UNC notation as `\\server\share\Labels\label.lbl` (where G: drive is mapped to `\\server\share`).

Notation For Accessing Files In Control Center

When opening a file in Document Storage inside Control Center, use the HTTP notation such as `http://servername:8080/label.lbl`, or WebDAV notation as `\\servername@8080\DavWWWRoot\label.lbl`.

Additional notes:

1. The user account that is used to run a service is used to obtain files from the Document Storage. This user must be configured in Control Center Administration tab in order to gain access to files in the Document Storage.
2. The WebDAV access can only be used with Windows user authentication in Control Center.

Printer Drivers Availability

To print labels using a network shared printer, make the printer driver available on the server where Designer is installed on.

Make sure the user account that Designer runs under has access to the printer driver. If the network printer was just installed on the machine, Designer might not see it until you restart the Service.

TIP: To allow automatic notification of new network printer drivers, you have to enable the appropriate inbound rule in Windows firewall. For more information, see [Knowledge Base article KB 265](#).

Search Order For Requested Files

When the Designer attempts to load a specified label or image file, it does not cancel the processing and reports an error in case the file is not found. It tries to locate the requested file at alternate locations.

Designer performs file location checks in the below listed order:

1. Check if the file exists at the location as defined in the action.
2. Check if the file exists in the same folder as the solution or label file.
3. Check if the label file exists in .\Labels folder (for graphic files check .\Graphics folder).
4. Check if the label file exists in ..\Labels folder (for graphic files check ..\Graphics folder).
5. Check if the file exists in the global Labels folder (Graphics folder for graphics files).

If the file cannot be found at any of above listed locations, the action fails. An error is raised.

NiceLabel Print

Print is a standalone application for fast and easy printing. It eliminates the need for opening label and solution documents in Designer.

Print window consists of:

- **File location selector:** drop down list lets you select and manage the locations that store labels or solutions.

See section below for more details on files and locations.

- **Search:** finds the requested document.
- **Location folder structure:** displays the folders that are selected in the **File location selector**.
- **Document display area:** presents the documents which are stored in the selected folder.

Managing Document Locations

When using the Print for the first time, a blank Print window appears. Click **Manage Locations** in the **File location selector**. **Manage Locations** dialog opens.

Use **Manage Locations** dialog to browse for document locations on your system or network.

- **Add:** button for adding the label files:
 - **Folder Location:** browses for files on your system or network.
 - **PowerForms Web/Cloud location:** opens an additional window for specifying the server that hosts the label or solution files.
 - **Server URL:** server location.

EXAMPLE PowerForms Web server location – `http://server/PowerFormsWeb`

- Insert **User name** and **Password** to connect to a protected server.

NOTE User name and password are optional. With enabled authentication, the user is prompted for credentials if the user name and password fields are left empty before opening a solution from server.

- **Move up** and **Move down:** change the order of selected label locations.
- **Delete:** removes the location from Print.

Opening The Documents

After defining the local or remote location that stores the documents, start with printing. Follow the [steps in this section](#) to successfully print the labels.

Online Support

You can find the latest builds, updates, workarounds for problems and Frequently Asked Questions (FAQ) on the product web site at www.nicelabel.com.

For more information, please refer to:

- Knowledge base: <http://www.nicelabel.com/support/knowledge-base>
- NiceLabel Support: <http://www.nicelabel.com/support/technical-support>
- NiceLabel Tutorials: <http://www.nicelabel.com/learning-center/tutorials>
- NiceLabel Forums: <http://forums.nicelabel.com/>

NOTE If you have a Service Maintenance Agreement (SMA), please contact the premium support as specified in the agreement.

Americas

+1 262 784 2456

sales.americas@nicelabel.com

EMEA

+386 4280 5000

sales@nicelabel.com

Germany

+49 6104 68 99 80

sales@nicelabel.de

China

+86 21 6249 0371

sales@nicelabel.cn

www.nicelabel.com

