

**SnapForm**  
Excellence in eForms™



## *Snapform Designer User Manual*



*Viewer*



*Designer*



*eForms*

# Table of contents

	<b>Table of contents</b>	<b>i</b>
<b>1</b>	<b>Introduction</b>	<b>1 - 1</b>
1.1	Welcome to the World of Snapform	1
1.2	About this manual	2
1.2.1	Users	2
1.2.2	Conventions	2
1.2.2.1	Representation in the Text	2
1.2.2.2	Screenshots	3
1.2.2.3	Remarks	3
1.2.3	Covered Software Versions	4
1.3	Snapform Components	5
1.3.1	Snapform PDF Converter	5
1.3.2	Snapform Designer	5
1.3.3	Snapform Viewer	5
1.3.4	Workflow and Management Tools	6
1.3.5	API and SDK	6
1.4	The QDF Document Format	7
1.4.1	General	7
1.4.2	QDF/S	8
1.4.3	QDF/A	9
<b>2</b>	<b>Installation</b>	<b>2 - 1</b>
2.1	System Requirements	1
2.2	Preparing the system	2
2.3	Standard Installation	2
2.3.1	Installing the Snapform Designer	3
2.3.2	Installing the Snapform Viewer	3
2.3.3	Installing the Snapform PDF Converter	4
2.4	Registering	5
2.5	Configuring the Snapform Designer	6
2.6	Uninstalling	6
<b>3</b>	<b>Using the Snapform Designer</b>	<b>3 - 1</b>
3.1	Starting and Exiting the Application	1
3.1.1	Starting the Application	1
3.1.2	Exiting the Application	2
3.2	Components of The Snapform Designer	3
3.2.1	The Application Window	3

3.2.2	Menu bar	4
3.2.2.1	The Apple Menu (Macintosh-specific)	4
3.2.2.2	The Snapform Designer Menu (Macintosh-specific)	4
3.2.2.3	The File Menu	7
3.2.2.4	The Edit Menu	9
3.2.2.5	The Format Menu	12
3.2.2.6	The Controls Menu	15
3.2.2.7	The View Menu	17
3.2.2.8	The Tools Menu	17
3.2.2.9	The Help Menu	18
3.2.3	Toolbar	19
3.2.4	Controls bar	23
3.2.5	Workspace/Editor window	23
3.2.6	Properties window	24
3.2.7	Tools window	25
3.2.7.1	Controls	25
3.2.7.2	Tools	25
3.2.7.3	Find and Replace	26
3.2.8	Expression editor	28
3.2.9	Other user interface components	30
3.2.9.1	Keyboard Shortcuts	30
3.2.9.2	Color picker	30
3.3	Operation modes of The Snapform Designer	33
3.3.1	Draft mode	33
3.3.2	Preview mode	33
4	Forms Development	4 - 1
4.1	Introduction	1
4.2	Analysis and planning	2
4.3	Base layout	4
4.3.1	Background images	4
4.3.1.1	Snapform PDF Converter	4
4.3.1.2	Conversion Service	4
4.3.2	The Snapform PDF Converter	5
4.3.2.1	Preparing the PDF documents	5
4.3.2.2	The user interface of the Snapform PDF Converter	7
4.3.2.3	Preferences	11
4.3.2.4	Converting a single document	12
4.3.2.5	Batch conversion	12
4.3.2.6	Conversion problems	14
4.3.3	Creating a new document	15

4.3.3.1	Importing existing background pages	16
4.3.3.2	Creating empty pages	17
4.3.3.3	Properties of new forms	19
4.3.4	Document pages	20
4.3.4.1	Adding and removing pages	21
4.3.4.2	Page properties	22
4.3.5	Graphical elements	26
4.3.5.1	Lines	26
4.3.5.2	Rectangles	33
4.3.5.3	Covers	39
4.3.5.4	Images and logos	45
4.3.6	Text	54
4.3.6.1	The Label tool	55
4.3.6.2	HTML tags and attribute usable in Snapform	63
4.4	Form layer	64
4.4.1	Labels and expression container	64
4.4.2	Entry fields	65
4.4.2.1	Entry fields in general	65
4.4.2.2	Text fields	72
4.4.2.3	Multiline fields	76
4.4.2.4	Numeric fields	79
4.4.2.5	Date fields	88
4.4.2.6	Masked input	93
4.4.2.7	Password fields	98
4.4.2.8	Examples of entry fields	101
4.4.3	Check boxes and Radio buttons	103
4.4.3.1	Check box	103
4.4.3.2	Radio buttons	108
4.4.3.3	Setting up radio buttons	112
4.4.3.4	Examples of Check boxes and Radio buttons	114
4.4.4	Selection lists	117
4.4.5	1D Barcodes	125
4.4.5.1	The Barcode tool	125
4.4.5.2	1D barcodes supported in Snapform	138
4.4.5.3	1D barcode examples	144
4.4.6	2D Barcodes	151
4.4.6.1	2D barcodes in general	151
4.4.6.2	Aztec barcode	152
4.4.6.3	PDF-417 Barcode	163
4.4.6.4	Datamatrix	174



4.4.6.5	QRCode	184
4.4.7	Tables	195
4.4.7.1	The Table tool	197
4.4.7.2	Assembling tables	207
4.4.7.3	Examples of tables	218
4.4.8	Active elements	222
4.4.8.1	Hyperlinks	222
4.4.8.2	Infopoints	229
4.4.8.3	Buttons	240
4.4.8.4	Applied active elements: A help system	240
4.5	Actions and logic	243
4.5.1	Carriers of action and logic	243
4.5.1.1	Fields	244
4.5.1.2	Functions	244
4.5.2	Actions and logic available in Snapform	245
4.5.2.1	String functions	245
4.5.2.2	Regular Expression functions	246
4.5.2.3	Math functions	247
4.5.2.4	Table functions	249
4.5.2.5	Date functions	250
4.5.2.6	State functions	252
4.5.2.7	Consistency check functions	253
4.5.2.8	Identifier functions	254
4.5.2.9	Combo box functions	255
4.5.2.10	Special functions	256
4.6	Integrity of the form	257
4.7	Stylesheets	258
4.7.1	The Stylesheet editor	258
4.7.2	Importing and exporting stylesheets	264
4.8	Form and security settings	265
4.8.1	Introduction	265
4.8.2	General	265
4.8.2.1	Document information	266
4.8.2.2	Templates	267
4.8.2.3	Printing	267
4.8.2.4	File format	268
4.8.3	Security	268
4.8.3.1	Passwords	269
4.8.3.2	Options	270
4.8.3.3	Action keys	272

4.8.4	Validation	274
4.8.4.1	Form expiry	275
4.8.4.2	Invalid form	275
4.8.4.3	Incomplete form	276
4.8.5	RDF Metadata	276
4.8.6	Server-side interaction	277
4.8.7	Fonts	279
4.8.8	eFiling	280
4.8.8.1	Format	281
4.8.8.2	File name	282
4.8.8.3	Target	282
4.8.8.4	UI settings	282
4.9	Release and deployment	283
4.10	Forms management	284
5	Expression language reference	5 - 1
5.1	Constants	3
5.1.1	TRUE	3
5.1.2	FALSE	3
5.1.3	pi	3
5.1.4	e	4
5.1.5	null	4
5.2	Operators	5
5.2.1	$x + y$	5
5.2.2	$x - y$	5
5.2.3	$x * y$	5
5.2.4	$x / y$	6
5.2.5	$x ^ y$	6
5.2.6	$x \% y$	6
5.2.7	$x = y$	7
5.2.8	$x != y$	7
5.2.9	$x >= y$	7
5.2.10	$x > y$	8
5.2.11	$x <= y$	8
5.2.12	$x < y$	9
5.2.13	$x \text{ AND } y$	9
5.2.14	$x \text{ OR } y$	9
5.2.15	NOT x	10
5.3	Functions	11
5.3.1	Abs(x)	11
5.3.2	Acos(x)	11

5.3.3	addDay(date_value, number)	12
5.3.4	addHour(time_value, number)	13
5.3.5	addMinute(time_value, number)	14
5.3.6	addMonth(date_value, number)	15
5.3.7	addYear(date_value, number)	16
5.3.8	Asin(x)	17
5.3.9	Atan(x)	18
5.3.10	calcDays(date_first, date_second)	18
5.3.11	calcHours(date_first, date_second)	19
5.3.12	calcMinutes(date_first, date_second)	20
5.3.13	Ceil(x)	21
5.3.14	Cell(table, column, row)	22
5.3.15	checkCardNumber(value)	23
5.3.16	checkMod10(value)	23
5.3.17	checkMod10r(value)	24
5.3.18	Compress(x)	25
5.3.19	Cos(x)	25
5.3.20	Cover(x)	26
5.3.21	createGUID()	27
5.3.22	Date(year, month, day)	27
5.3.23	Exp(x)	28
5.3.24	find(string, regex[, start] )	29
5.3.25	Floor(x)	30
5.3.26	getBytes(string)	30
5.3.27	getCol()	31
5.3.28	getContent(url)	32
5.3.29	getDataID()	32
5.3.30	getDay(date_value)	33
5.3.31	getHour(time_value)	34
5.3.32	getMinute(time_value)	34
5.3.33	getMod10(value)	35
5.3.34	getMod10r(value)	35
5.3.35	getMonth(date_value)	36
5.3.36	getPrintID()	37
5.3.37	getPrintIdStr(length [, digits]	37
5.3.38	getRandomString(length [, digits])	38
5.3.39	getRow()	39
5.3.40	getSelectedIndex(ComboBox)	39
5.3.41	getSessionID()	40
5.3.42	getValues(delimiter, value1 [, value2[, ...]])	41

5.3.43	getVersion()	42
5.3.44	getYear(date_value)	42
5.3.45	if(logical_test, value_if_true [,value_if_false])	43
5.3.46	importData(url)	44
5.3.47	IncVersion(initial_value, step)	45
5.3.48	InStr(string, pattern [, start_position])	45
5.3.49	isEmpty(x)	47
5.3.50	Left(string, count)	47
5.3.51	Len(x)	48
5.3.52	Log(x)	49
5.3.53	match(string, regex)	49
5.3.54	Max(x, y)	50
5.3.55	Min(x, y)	51
5.3.56	Now()	51
5.3.57	replaceAll(orig_string, regex, replace_with)	52
5.3.58	Right(string, count)	53
5.3.59	Round(x)	54
5.3.60	setFieldPrintable(true false, field_name)	54
5.3.61	setFieldVisible(true false, field_name)	55
5.3.62	setFieldWritable(true false, field_name)	56
5.3.63	setList(ComboBox, str_items, str_values)	56
5.3.64	setPagePrintable(true false, page_number [, page_number[,...]])	58
5.3.65	setPageVisible(true false, page_number [, page_number[, ...]])	58
5.3.66	setPrintable(category_list, true false)	59
5.3.67	setSelectedIndex(ComboBox, index)	60
5.3.68	setVisible(category_list, true false)	61
5.3.69	setWritable(category_list, true false)	61
5.3.70	Sin(x)	62
5.3.71	Sqrt(x)	63
5.3.72	Str(x)	63
5.3.73	StrTotal(table, column)	64
5.3.74	SubArray(byte_array, start [, count])	65
5.3.75	SubStr(string, start[, count])	66
5.3.76	Tan(x)	67
5.3.77	TblValues(table, tbl_start, tbl_end, row_start, row_end, delimiter, column [, column[, ...]])	67
5.3.78	Time(hour, minute[, second])	69
5.3.79	TimeTotal(table, column)	70

5.3.80	Today()	70
5.3.81	Total(table, column)	71
5.3.82	TrimStr(string)	72
5.3.83	Trunc(x)	72
5.4	Regular Expressions	74
5.4.1	Overview table	74
6	Snapform File Formats	6 - 1
6.1	Form, compressed (QDF/S)	1
6.2	Form, open (QDF/A)	1
6.3	Form Template, compressed (QDF/S)	4
6.4	Form Template, open (QDF/A)	4
6.5	Background layer	5
6.6	Field layer	5
6.7	Data, binary	5
6.8	Data, open	6
6.9	The Snapform Schema	6
7	API and Toolkits	7 - 1

# 1 Introduction

## 1.1 Welcome to the World of Snapform

Snapform is a comprehensive system for developing, managing, and interacting with electronic forms.

The Suite consists of several components: The Snapform Designer is the development tool which is used to create the forms. The Snapform Viewer is the client component for filling out and interacting with the forms. The Snapform Workflow and Management Tools help managing forms and data.

Snapform uses the QDF file format which is based on XML technologies. Data is represented in XML and can be processed in a simple and straightforward manner.

Snapform allows an electronic version of the form to be integrated into paper-based workflows. The addition of barcodes can help streamline these processes.

The forms developer is supported by a unique palette of entry objects and predefined expressions. The Snapform Viewer is available in more than 32 different languages, which makes it an excellent choice for deploying forms in a global environment.

## 1.2 About this manual

### 1.2.1 Users

This manual is designed for forms managers, forms developers, as well as individuals responsible for electronic forms who plan, create, maintain, and distribute Snapform forms. This manual provides sufficient information for anyone working with Snapform forms to successfully fulfill their tasks.

### 1.2.2 Conventions

#### 1.2.2.1 Representation in the Text

This manual uses various display conventions.

Menu names are displayed **in this font**.

Buttons are represented either as an icon, or with their name represented **in this font**.

Texts of GUI-Elements, such as dialogs, error messages, etc. are represented **in this font**.

Keyboard keys are represented as in this example: **<Ctrl><C>**. Note that the characters engraved on the keyboard may differ, depending on the origin and localization of the keyboard.

Keyboard entries are displayed **in this font**.

System responses in interactive dialogs are represented **in this font**.

Elements of expressions (such as function names) or file names are represented **in this font** if they are part of body text.

Placeholders for field names and variables are shown in the body text and program listings like this: **<field name>**. In order to make such

expressions work, these placeholders must be replaced with actual field names..

Expressions and program code in listings are represented **in this font**.

### 1.2.2.2

### Screenshots

Screenshots help you find your way around the screen, represent elements of the user interface, as well as to visually represent examples. The Snapform Designer is platform independent, and it uses functions of the operating system for GUI elements, it is possible that the actual display on screen differs from the images shown in this manual. This manual uses screenshots from a Windows and from a Mac OS X environment in a non-specific manner.

When platform-specific features are shown, they are clearly indicated.

**Note:** *In some cases, it is possible that screenshots have been edited to make certain features more clear.*

It is assumed that users of the Snapform Designer are familiar with their computer, and understand the basic operation of their operating system, as well as typical applications. In Step-by-Step instructions, screenshots will only be shown when necessary.

### 1.2.2.3

### Remarks

Remarks are labelled depending on their importance as "Attention", "Note" and "Helpful Hint".

**Attention:** *A remark labelled "Attention" contains information which is important, and must be considered, as it might otherwise lead to serious malfunction or data loss.*

**Note:** *A remark labelled "Note" contains explanatory information or additions to the text. It is highly recommended to follow these notes.*



**Helpful Hint:** *A remark labelled “Helpful Hint” contains information which makes working with the program easier, and is based on practical experience.*

### 1.2.3

## Covered Software Versions

This manual covers the behavior of the Snapform Designer version 5.1.9 and may refer to the Snapform Viewer version 1.5.24. It can be used for older versions, but certain features may not be present.

## **1.3 Snapform Components**

### **1.3.1 Snapform PDF Converter**

The format converter converts PDF documents into the Snapform file format. This prevents the need for recreating the layout when working with legacy forms. The forms developer can then concentrate on the implementation of the form's functionality and logic, and save a considerable amount of time.

This application is available for Windows only. As an alternative to using the Snapform PDF converter, Ringler Informatik AG offers a conversion service.

The converter is explained in detail in section 4.3.2.

### **1.3.2 Snapform Designer**

The Snapform Designer is the development tool for Snapform forms. This application is based on the Java Runtime Engine and is available for Microsoft Windows and the Apple Mac OS X.

This manual describes the functionality of the Snapform Designer

### **1.3.3 Snapform Viewer**

The Snapform Viewer is the client component for Snapform forms, and can be downloaded free from the Snapform web site. It is based on the Java Runtime Engine and is available for Windows, Macintosh and various Unix/Linux variants.

The use of the Snapform Viewer is self-explanatory.

## **1.3.4 Workflow and Management Tools**

The Workflow and Management Tools extend Snapform to a complete eForms solution and allow a centralized data management as well as a centrally controlled security policy for the forms and their data.

The Workflow and Management Tools are not covered by this manual. For further information, contact Ringler Informatik AG.

## **1.3.5 API and SDK**

For Snapform-based applications, an API with an according SDK exists. These components are not covered in this manual. For further information, contact Ringler Informatik AG.

## 1.4 The QDF Document Format

### 1.4.1 General

Snapform's document format is called QDF (Quick Document Format), and is based on XML. QDF consists essentially of the following five components:

- A reduced set of SVG to display background layouts and graphics
- A mechanism for embedding and replacing fonts
- An object layer with vector-based data entry components
- A data layer representing data in a structured manner
- A structured storing system to bundle these elements and attachments into one single file, where data can be optionally compressed

The sum of these five components leads to important advantages:

- QDF documents are logically preformatted and therefore extremely compact in size
- Form Fields and business logic are also very compact, and increase the documents size only by a small percentage
- The SVG-based background layer of the form provides printing formats in vector quality and also guarantees a complete representation of transparency
- The font replacement strategy of Snapform allows a platform-independent original representation of fonts, even if the particular font is not installed on the user's computer

Other features of QDF/Snapform:

- The precise representation of a paper-based form layout on screen and when printed, permits the use of this technology in document-based workflows, where both data and their presentation are important.
- Support for all current barcode symbologies (1D and 2D) reduce breaks in workflows and allow at the same time the intergration with existing legacy business applications.
- Support for a wide range of text entry methods (including Chinese, Japanese, Korean, Hebrew and Arabic) allows world-wide use of the forms.
- The user interface of the Snapform Viewer is available in more than 32 languages and thus supports general public users with ease.

The licensing model of Snapform contains an unrestricted and free of charge use of Snapform forms. The form user does not encounter limitations which can only be overcome or enabled with additional software (or services).

A more in-depth description of the QDF file format (as far as it has been disclosed outside of special developments) can be found in chapter 6.

QDF can be formatted either compressed/encrypted or as an open XML format:

## 1.4.2

### QDF/S

QDF/S is the most thoroughly protected and most compact variant of QDF. All components are compressed and encrypted (128 bit), and can only be opened with the Snapform Viewer and properly equipped server applications.

### 1.4.3

## QDF/A

QDF/A is the “standard” version of QDF where the data layer is disclosed in XML format. This allows archival or document management software to access and manage filled-out Snapform forms without the need for additional tools.

# 2 Installation

## 2.1 System Requirements

In order for Snapform to be installed and to run properly, a few system requirements must be fulfilled.

### Platform

The Snapform Viewer will run on any platform (Windows, Macintosh, Unix), where Sun Java Runtime 1.4 or newer is running. This means for Windows: Windows ME, Windows 2000, Windows XP, Windows Vista. For Macintosh, OS X 10.1 and newer.

For the Snapform Designer, there are some limitations which do not allow for Linux support. For the Snapform Designer, either Windows (variants see above) or Mac OS X 10.2.8 or newer are required.

The small “footprint” of the Snapform Viewer and the Designer do not require a specific amount of memory and disk space. Under low memory conditions, the working speed will be limited.

The Snapform PDF Converter requires Windows 2000, Windows XP or Windows Vista. Intel-based Macintoshes with virtual machines running one of the mentioned Windows variants will work as well.

### Other Hardware

Further hardware consists, if paper-based workflows are planned, of a printer with a resolution of at least 300 dpi. If barcodes with small module sizes are used, 600 dpi or higher is recommended.

For applications using PKI, a signature card reader with touchpad may be necessary.

On the back-end side for paper-based workflows, readers for barcodes may be needed. The performance of these devices depends on the amount of forms to be processed.

## System Software

As mentioned above, the Snapform Viewer and the Snapform Designer require Java Runtime Engine 1.4 or newer.

Further system software is not required.

For electronic data transmission of forms and forms data to a centralized back-end system, an active internet connection is required. This requirement is application-dependent.

## 2.2 Preparing the system

A prerequisite for the installation of the Snapform Viewer or the Snapform Designer is the Sun Java Runtime Engine version 1.4 or newer. These components may already be part of your operating system, or they can be downloaded from the Sun website (<http://www.java.com>) for Windows or from the Apple website (<http://www.apple.com>) for Macintosh.

## 2.3 Standard Installation

The installation procedure for the Snapform Designer and the Snapform Viewer is simple and straightforward due to a guided and easy to follow installation routine. When the Java Runtime Engine is not available on a Windows system, it will automatically be downloaded and preconfigured by the installer, or it can be downloaded from the Sun website (<http://www.java.com>).

When both the Snapform Designer and the Snapform Viewer are installed, it is recommended to first install the Designer, and then the Viewer. The installation order only has an effect on the filetype assignments.



## 2.3.1

### Installing the Snapform Designer

The most recent installer of the Snapform Designer can be downloaded from the Snapform website (<http://www.snapform.com>).

Note that the Snapform Designer can be installed with User privileges, and does not require any Administrative privileges.

#### Windows

The Windows version of the Snapform Viewer is a **.exe** file. After downloading, the installation wizard can be started at any time by double-clicking on the **.exe** file. The installation steps are defined in the wizard, and are self-explanatory.

When the installation routine has successfully completed, the Snapform Designer is ready to run.

#### Macintosh

The Macintosh version of the Snapform Designer is available as disk image in the **.dmg** format. After mounting the disk image, double-click the installer icon to start the installation. The installation steps are self-explanatory.

When the installation routine has successfully completed, the Snapform Designer is ready to run.

## 2.3.2

### Installing the Snapform Viewer

The most recent installer of the Snapform Viewer can be downloaded from the Snapform web site (<http://www.snapform.com>).

Note that the Snapform Viewer can be installed with User privileges, and does not require any Administrative privileges.

#### Windows

The Windows version of the Snapform Viewer is a **.exe** file. After downloading, the installation wizard can be started at any time by double-clicking the **.exe** file. The installation steps are defined in the wizard, and are self-explanatory.

When the installation routine has successfully completed, the Snapform Viewer is ready to run.

## Macintosh

The Macintosh version of the Snapform Viewer is available as disk image in the **.dmg** format. After mounting the disk image, double-click the icon to start the installer. The installation steps are self-explanatory.

When the installation routine has successfully completed, the Snapform Viewer is ready to run.

## Linux

The Linux version of the Snapform Viewer is available as a shell installer. After downloading the file, launch the installer shell script. The installation steps are self-explanatory.

**Note:** *The installer attempts to write to the **/usr/local** directory. It may therefore be necessary to run the installer script with raised privileges. using the **sudo** command.*

When the installation routine has successfully completed, the Snapform Viewer is ready to run.

### 2.3.3

## Installing the Snapform PDF Converter

The Snapform PDF Converter installer is a **.exe** file. After downloading, the installation wizard can be started at any time by double-clicking the **.exe** file. The installation steps are defined in the wizard, and are self-explanatory.

When the installation routine has successfully completed, the Snapform PDF Converter is ready to run.

**Note:** *The Snapform PDF Converter is a .NET application. When a newer version of the Snapform PDF Converter has to be installed, you must first remove older versions using the Snapform PDF Converter's uninstaller.*

## 2.4 Registering

The Snapform Viewer can not be registered. This section is therefore relevant for the Snapform Designer only.

A recently downloaded and installed version of the Snapform Designer will be launched in Demo mode.


In order to activate the licensed functionality, the license key must be entered. The license key is normally sent to you by Ringler Informatik. The key also contains information about the licensed function range and the number of users.

After selecting the menu item **Enter license key** in the **Help** menu, the window to enter the license key opens (see Fig. 2-1).

**Fig.2-1** *Window to enter the license key*

Enter the license information in this window. Note that the values in **User name** and **Company name** must be the same as specified in the order for the license key.

The **License level** field is filled out automatically when the license key is validated.

The license key is easiest entered by loading the license file. The Browse icon,  located to the right of the entry field for the license key, opens an "Open File" dialog, where the license file can be specified. After

confirming the entry, the file opens and the key is registered automatically.

It is possible that the license key has already been communicated in another way (such as a text string as part of an e-mail message). In this case, it is recommended to copy the key string onto the clipboard and then paste it by clicking on the **Paste from clipboard** button.

Confirm the entries with **OK**. After the data is verified, the Snapform Designer is activated according to the defined product level.

## 2.5 Configuring the Snapform Designer

The Snapform Designer does not require any further configuration.

The various language versions are named **Snapform Designer EN** (English), **Snapform Designer DE** (German) and **Snapform Designer FR** (French) and can be specifically opened as follows:

### Windows

**Start** Menu —> **all Programs** —> **Snapform**, then select language version.

### Macintosh

The installation folder contains icons for the different language versions. These icons can be dragged into the dock or started from the Applications folder by double-clicking on them.

## 2.6 Uninstalling

The installation directory of the Snapform Designer and the Snapform Viewer contain an uninstaller.

To uninstall either application, launch the uninstaller. This will remove all relevant components of that application.

## 3 Using the Snapform Designer

### 3.1 Starting and Exiting the Application

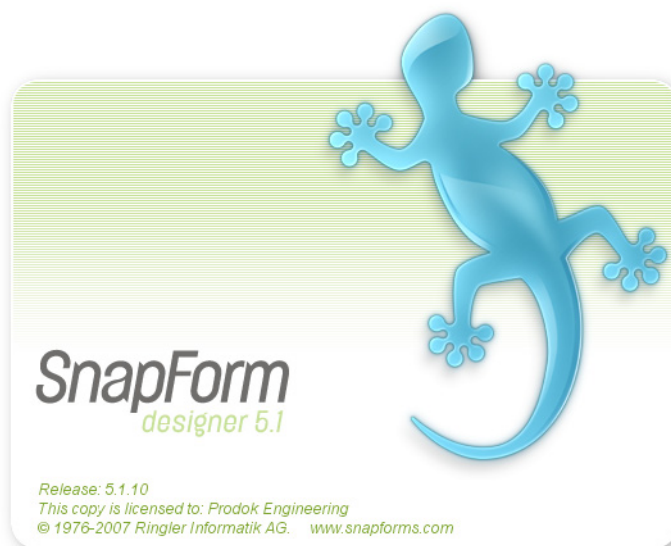
#### 3.1.1 Starting the Application

The Snapform Designer is started like any other application. This means that it is started from either the **Start** menu (Windows) or the Dock (Macintosh), or by double-clicking on the program icon or the icon of a file assigned with the Snapform Designer.

**Note:** When the Snapform Viewer is also installed on the system, it is possible that double-clicking a file of the type **.qdf** or **.qdft** may open the Snapform Viewer, depending on the file type associations. In such a case, the Snapform Designer can not be started by double-clicking such a file icon.

While the application is starting up, the start-up screen (Fig. 3-1) appears.

**Fig.3-1** Start-up screen



The start-up screen shows the version number, the registration information and the manufacturer of the software.

When the application has fully loaded, the workspace appears (see section 3.2.1).

When the start-up process has completed, an empty application window appears, which is described below (section 3.2.1).

## 3.1.2

### Exiting the Application

The application is normally exiting using the menu item **File —> Quit** (Windows), **Snapform Designer —> Quit** (Macintosh). If there are unsaved documents, a popup will ask whether you want to save them.

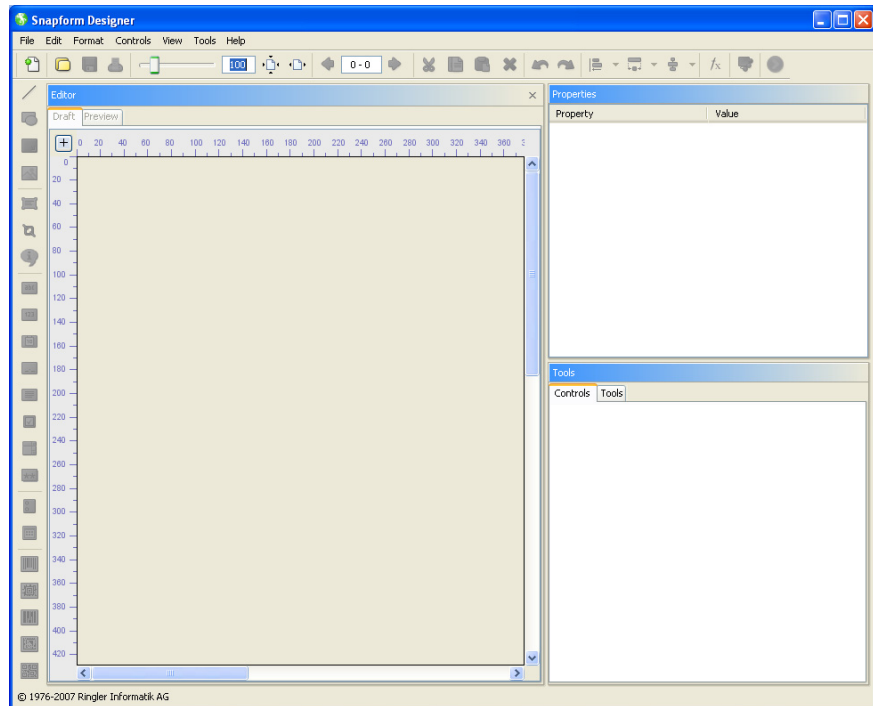
The application can also be exited using a keyboard shortcut (**<Alt><F4>** on Windows, **<Ctrl><Q>** on Macintosh) or the closing boxes.

## 3.2 Components of The Snapform Designer

### 3.2.1 The Application Window

The empty application window is shown in Fig. 3-2.

**Fig.3-2** *Empty Application window*



The application window consists of the following areas:

- 1 Menu bar and menus
- 2 Toolbar
- 3 Controls bar
- 4 Workspace
- 5 Properties window
- 6 Tools window

These components are explained in the following sections.

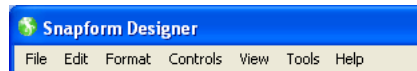
## 3.2.2

## Menu bar

The menu bar contains the most important functions of the Snapform Designer. The menu items are context-dependently active. Inactive elements are greyed out and can not be selected.

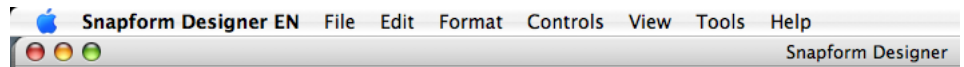
The menu bar is platform-dependent. The Windows version is shown in Fig. 3-3, and the Macintosh version in Fig. 3-4.

**Fig.3-3** *The menu bar of the Windows version*



The menu bar of the Windows version is a subset of the Macintosh menu bar. The Apple menu and the menu **Snapform Designer** are not present.

**Fig.3-4** *The menu bar of the Macintosh version*



### 3.2.2.1

### The Apple Menu (Macintosh-specific)

The Apple menu does not contain any menu items relevant for the Snapform Designer.

### 3.2.2.2

### The Snapform Designer Menu (Macintosh-specific)

The **Snapform Designer** menu (see Fig. 3-5) is specific for the Macintosh implementation and contains application-dependent menu items provided by the system:

**Fig.3-5** *The Snapform Designer menu*





**Note:** For these screen shots, the specific English language version was used. This menu item is therefore called **Snapform Designer EN**.

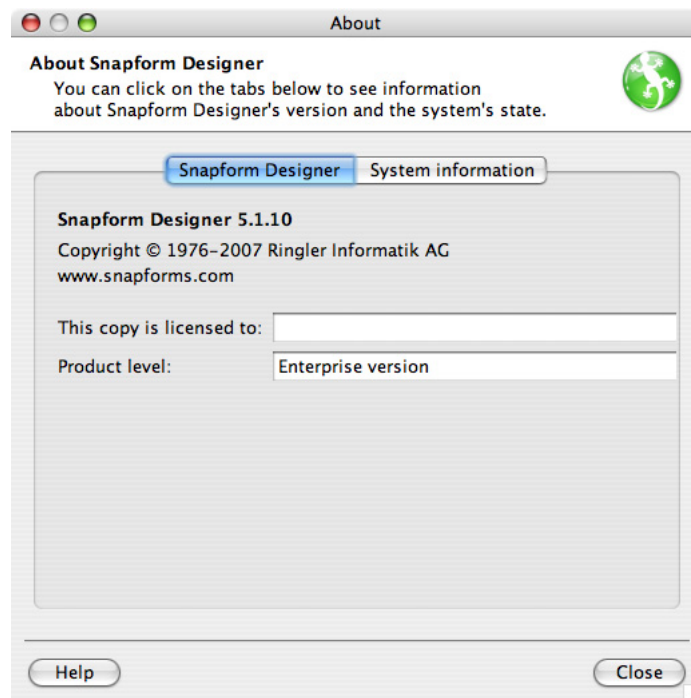
### About Snapform Designer (Macintosh only)

Selecting this menu item opens an information window about the Snapform Designer.

The menu item **About Snapform Designer** shows the current version information as well as the license information (see Fig. 3-6).

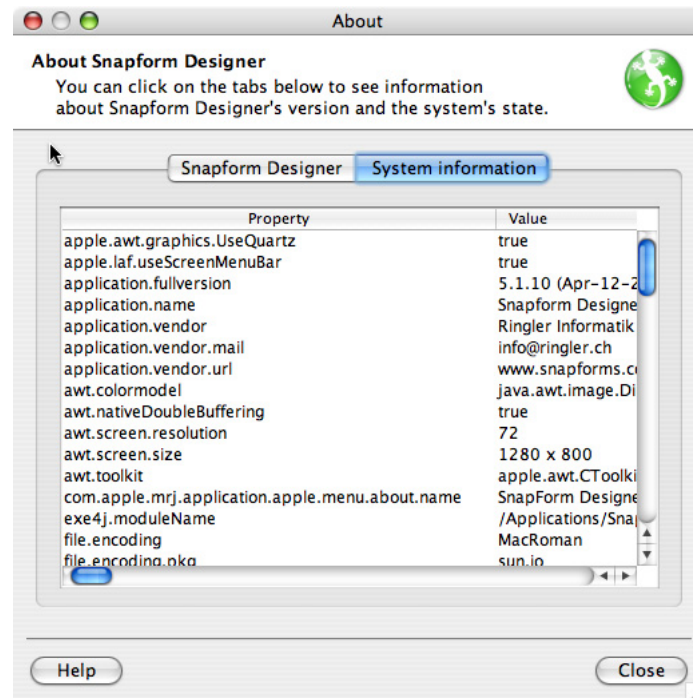
**Note:** Under Windows, this menu item is located in the **Help** menu.

**Fig.3-6** The About Snapform Designer window



The **System information** display shows an overview over the system environment relevant to the Snapform Designer (see Fig. 3-7). This information can be useful for debugging purposes.

**Fig. 3-7** *The About System information window*



## Preferences

This is a placeholder provided by the operating system to be used for a general preferences window. Snapform Designer does not have any user-configurable options (some base settings are automatically set depending on the context).

## Services

This is a Macintosh specific menu item which provides access to the various services provided by the operating system and other installed applications. The options of this menu item are installation-dependent and don't have a direct relationship to the Snapform Designer.

## Hide Snapform Designer

Hides all Snapform Designer windows.

## Hide Others

Hides all other running applications except the Snapform Designer.

## Quit Snapform Designer

Closes the Snapform Designer

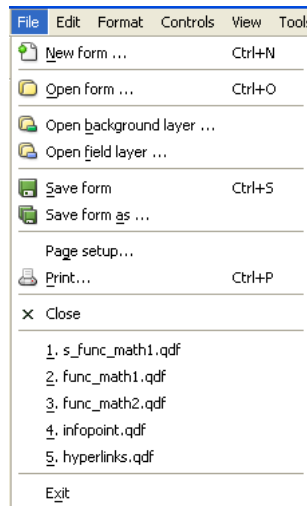
**Note:** Under *Windows*, the command to exit the Snapform Designer is located in the **File** menu.

### 3.2.2.3

## The File Menu

The **File** menu (see Fig. 3-8) contains file-oriented commands, such as opening and saving files.

**Fig. 3-8** *The File Menu*



### New Form

Opens the specification window for a new form. This function is explained in section 4.3.3.

### Open Form

Opens an “Open File” dialog where the requested **.qdf** document is selected.

If there is already a document open, it will be closed first. If there are unsaved changes, a dialog appears asking whether the document should be saved (see Fig. 3-10).

### Open Background layer

Opens an “Open File” dialog from which the background layer document (in the **.hdt** format) you are searching for can be selected.

The background layer is created in a separate step, where a PDF document is converted into a .hdt file using Snapform PDF Converter.

If there is a document currently open, it will be closed first. If there are unsaved changes, a dialog appears asking whether the document should be saved (see Fig. 3-10).

## Open Field layer

Opens an “Open File” dialog from which the background layer document (in the **.hdo** format) can be selected.

If there is already a document open, it will be closed first. If there are unsaved changes, a dialog appears, asking whether the document should be saved (see Fig. 3-10).

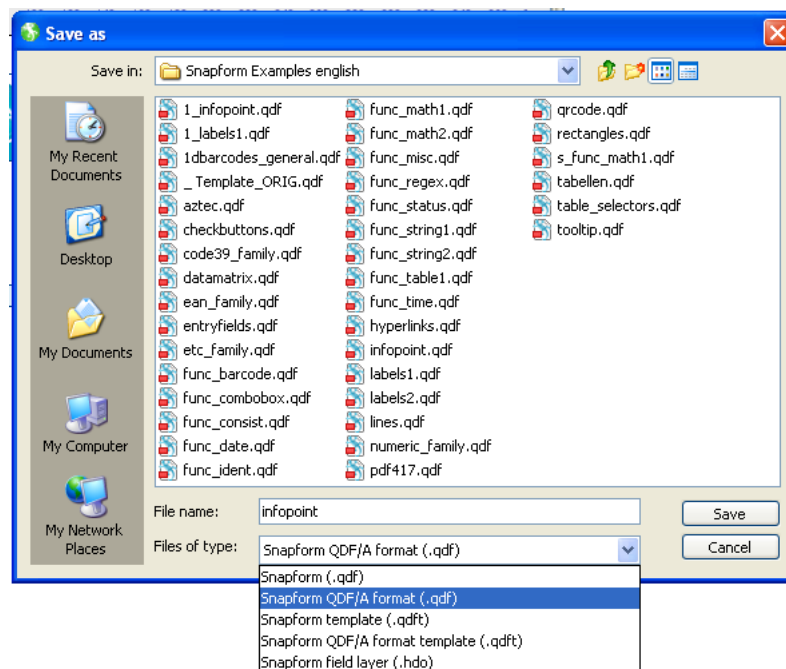
## Save Form

Saves the form under its existing name. An older version of the file will be overwritten.

## Save Form as...

Opens a “File Save” dialog where the target directory and the file name can be entered/selected (see Fig. 3-9).

**Fig.3-9** *The Save as... dialog*



The following file formats are available:

- **Snapform (.qdf)**: the standard Snapform format
- **Snapform QDF/A (.qdf)**: the standard Snapform file format with disclosed data layer

- **Snapform Template (.qdft):** Form template which always opens as an empty form
- **Snapform QDF/A Template (.qdft):** Form template with disclosed data layer which always opens as an empty form
- **Snapform Field layer (.hdo):** Format containing only the field layer

If no file extension is entered, it will be added automatically.

## Page Setup

Opens the normal “Page Setup” dialog which is provided by the operating system.

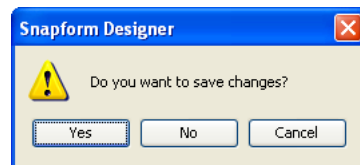
## Print

Opens the normal “Print” dialog provided by the operating system.

## Close

Closes the form. If there are unsaved changes, a dialog appears asking whether the document should be saved (see Fig. 3-10).

**Fig.3-10** *The Close document dialog*



### 1. to 5. last opened files

This list contains the 5 most recently opened files. Only files which have been opened using the **File open** dialog are placed into this list.

## Quit (Windows only)

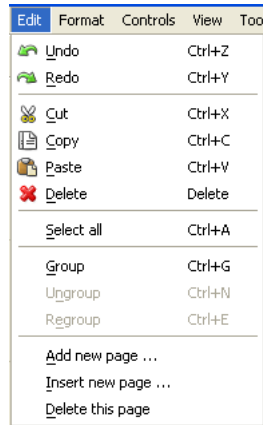
Closes the Snapform Designer. This command is in this menu under Windows only. On Macintosh, this command is located in the **Snapform Designer** menu.

### 3.2.2.4

## The Edit Menu

The **Edit** menu contains commands to edit elements (copy, align, etc.) (see Fig. 3-11).

**Fig.3-11** *The Edit Menu*



## Undo

The most recent action is undone

## Redo

The last action is repeated (also undoes the **Undo** command)

## Cut

The selected elements are copied to the clipboard and removed from the workspace. Form elements retain all their properties (except tab order).

## Copy

The selected elements are copied to the clipboard and remains in the workspace. Form elements retain all their properties (except tab order).

## Paste

The contents of the clipboard are pasted onto the workspace (as far as this is possible). All properties retained when placing the elements into the clipboard are applied. That means the positions of the elements are the same as they were when the elements were copied.

## Delete

The selected elements are deleted from the workspace without being placed into the clipboard.

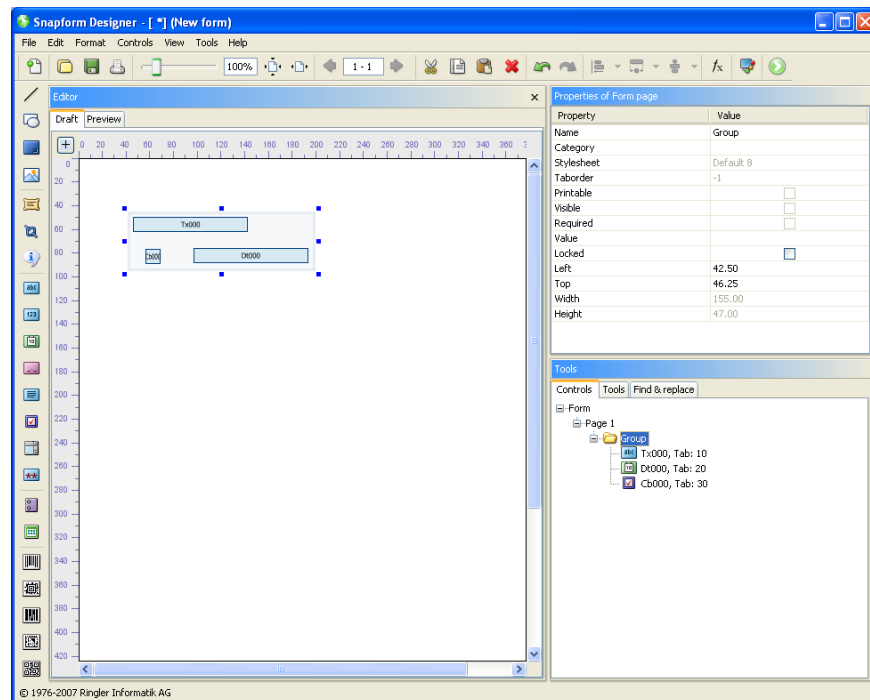
## Select all

All elements on the workspace are selected.

## Group

The selected elements are ordered into a group. The group gets a default name **Group** and can be treated as a single form element (see Fig. 3-12).

**Fig.3-12** *A group*



A Group is a form element into which other form elements can be placed. The main purpose of a group is to freeze the relative position of the grouped form elements. The following properties can be changed for a group.

## Name

The name of the group. This name can be changed.

## Locked

When this check box is selected, the group gets locked and can not be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

### **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the group.

### **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the group.

All other properties displayed in the Properties window cannot be changed.

Elements contained in a group can be selected in the object structure. This allows you to change their properties. However, the positional properties (**Left, Top, Width, Height**) are blocked. These properties become available only after the group has been dissolved with the **Ungroup** command.

### **Ungroup**

If the selected element is a group, this command dissolves the Group into its parts.

### **Regroup**

This command allows to expand the selected group with additional elements being part of the selection.

### **Add new page**

Adds a new page at the end of the document. This function is explained in section 4.3.4.1.

### **Insert new page**

Adds a new page in front of the current page. This function is explained in section 4.3.4.1.

### **Delete this page**

Removes the current page. This function is explained in section 4.3.4.1.

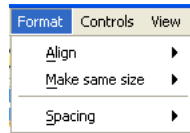
### **3.2.2.5**

#### **The Format Menu**

The Format menu contains functions to align selected form elements (see Fig. 3-13).



**Fig.3-13** *The Format Menu*

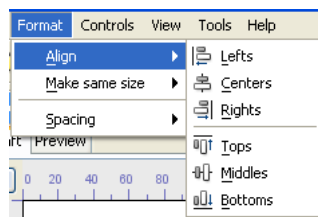


This menu is only active if at least two form elements are selected. It contains three submenus. The active selected element (with the full black markers on its bounding box) is the reference element.

### Submenu **Align**

This submenu (see Fig. 3-14) aligns the selected elements in relationship to each other. The alignment occurs according to the respective bounding box of the selected elements.

**Fig.3-14** *The menu Format  
—> Align*



The upper three items control the horizontal alignment, and the lower three items control the vertical alignment.

### **Left**

The elements are aligned horizontally to their left borders.

### **Center**

The elements are aligned horizontally to their centers.

### **Right**

The elements are aligned horizontally to their right borders.

### **Top**

The elements are aligned vertically to their top borders.

### **Middle**

The elements are aligned vertically to their centers.

### **Bottom**

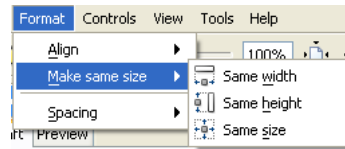
The elements are aligned vertically to their bottom borders.

Submenu

## Make same size

This submenu (see Fig. 3-15) adjusts the size of the selected elements in relationship to each other. The alignment occurs according to the respective bounding box of the selected elements.

**Fig.3-15** The menu *Format*  
—> *Make same size*



## Same width

All selected elements get the same width as the reference element.

## Same height

All selected elements get the same height as the reference element.

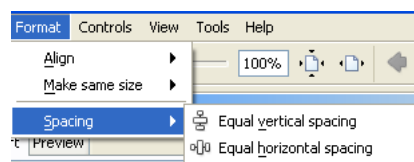
## Same size

All selected elements get the same width and height as the reference element.

Submenu **Spacing**

With the commands of this submenu (see Fig. 3-16), the spacing between the selected elements is adjusted. The adjustment occurs between the horizontal or vertical center axis of the bounding box of the elements. In order for this submenu to be active, at least three elements must be selected.

**Fig.3-16** The *Format* —>  
*Spacing* menu



## Equal vertical spacing

The vertical spacing between the horizontal center lines of the selected elements are changed to be equal.

## Equal horizontal spacing

The horizontal spacing between the vertical center lines of the selected elements are changed to be equal.

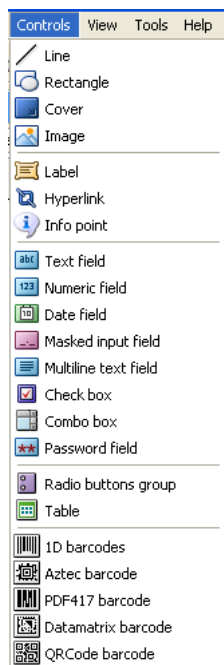
### 3.2.2.6

## The Controls Menu

**Note:** The term “Controls” is used in various standards concerning form elements. In this manual, the term “tool” may also be used.

The **Controls** (see Fig. 3-17) contains all the tools to create form elements.

**Fig.3-17** The Controls Menu



The list of the menu elements is the same as in the vertical tool bar (see 3.2.4).

### Line

The Line tool is explained in section 4.3.5.1.

### Rectangle

The Rectangle tool is explained in section 4.3.5.2.

### Cover

The Cover tool is explained in section 4.3.5.3.

### Image

The image tool is explained in section 4.3.5.4.

<b>Label</b>	The Label tool is explained in section 4.3.6.1.
<b>Hyperlink</b>	The Hyperlink tool is explained in section 4.4.8.1.
<b>Info point</b>	The Info point tool is explained in section 4.4.8.2.
<b>Text field</b>	The Text field tool is explained in section 4.4.2.2.
<b>Numeric field</b>	The Numeric field tool is explained in section 4.4.2.4.
<b>Date field</b>	The Date field tool is explained in section 4.4.2.5.
<b>Masked input field</b>	The Masked input field tool is explained in section 4.4.2.6.
<b>Multiline text field</b>	The Multiline text field tool is explained in section 4.4.2.3.
<b>Check box</b>	The Check box tool is explained in section 4.4.3.1.
<b>Combo box</b>	The Combo box tool is explained in section 4.4.4.
<b>Password field</b>	The Password field tool is explained in section 4.4.2.7.
<b>Radio buttons group</b>	The Radio buttons group tool is explained in section 4.4.3.2.
<b>Table</b>	The Table tool is explained in section 4.4.7.1.
<b>1D barcodes</b>	The 1D barcodes tool is explained in section 4.4.5.1.
<b>Aztec barcode</b>	The Aztec barcode tool is explained in section 4.4.6.2.
<b>PDF417 barcode</b>	The PDF417 barcode tool is explained in section 4.4.6.3.

## Datamatrix barcode

The Datamatrix barcode tool is explained in section 4.4.6.4.

## QRCode barcode

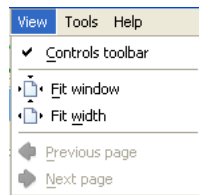
The QRCode barcode tool is explained in section 4.4.6.5.

### 3.2.2.7

## The View Menu

The View menu (see Fig. 3-18) contains functions about displaying the form on the workspace and for page navigation.

**Fig.3-18** *The View Menu*



## Controls toolbar

This menu item shows or hides the Controls toolbar (see section 3.2.3).

## Fit window

Scales the view of the active document so it fits completely into the Editor window.

## Fit width

Scales the view of the active document so it fits the width of the Editor window.

## Previous page

Switches to the previous page (only active when the document consists of multiple pages).

## Next page

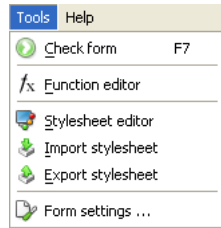
Switches to the next page (only active when the document consists of multiple pages).

### 3.2.2.8

## The Tools Menu

The **Tools** menu (see Fig. 3-19) contains special functions which are important when working with forms.

**Fig.3-19** *The Tools Menu*



## Check form

Checks the form for inconsistencies and incorrect expression syntax, and generates an error report (see Section 4.6).

## Function editor

Opens the Function editor. The Function editor is explained in section 4.5.1.2.

## Stylesheet editor

Opens the Stylesheet editor. The Stylesheet editor is explained in section 4.7.1.

## Import stylesheets

Opens an “Open file” dialog to select a form file (**.qdf**) or a stylesheet file (**.qdl**), from which the stylesheets will be imported (see section 4.7.2).

## Export stylesheets

Opens a “Save file” dialog and exports the stylesheets of the active document into a stylesheet file (**.qdl**) (see section 4.7.2).

## Form settings

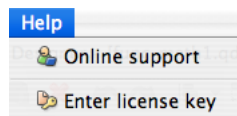
Opens the **Form settings** window (see section 4.8) to set various form and security options for the active form.

### 3.2.2.9

## The Help Menu

The Help menu (see Fig. 3-20) contains functions about using the Snapform Designer.

**Fig.3-20** *The Help Menu*



## Online support

This is the link to the support page on the Snapform website (<http://support.snapform.com/help/designer.asp?language=EN>)

## Enter license key

Opens the window for entering the license key (see section 2.4)

## About Snapform Designer (Windows only)

This menu item opens a window containing information about the Snapform Designer. This window has two selectable options. The **Snapform Designer** window displays the version information of the installed program and the licensing information (see Fig. 3-6).

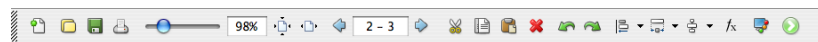
The **System information** window displays an overview of the system environment relevant to the Snapform Designer (see Fig. 3-7). This information can be useful for debugging purposes.

## 3.2.3

## Toolbar

The toolbar (see Fig. 3-21) contains tool icons for the most important actions with forms. The majority of the tool buttons are also available as menu action.

**Fig.3-21** *The Toolbar*



The individual tools are explained below.

### Tear off bar

The toolbar can be “torn off” the window and then appear as its own window on screen. When this window gets closed, the toolbar will reappear in the main window.

### New Form

This icon opens the specification window for a new form. This function is explained in section 4.3.3.

### Open Form

Opens an “Open file” dialog where the requested **.qdf** document is selected.

If there is already a document open, it will be closed first. If there are unsaved changes, a dialog appears asking whether the document should be saved (see Fig. 3-10).

### Save Form

Saves the form under its existing name. An older version of the file will be overwritten.

### Print Form

Opens the normal print dialog, provided by the operating system.

### Zoom slider



This icon allows you to set the zoom factor for displaying the form in the Editor window. Moving the marker changes the numeric zoom indicator, and the form will be displayed accordingly.

It is also possible to directly enter a percentage value. The zooms range is adjustable from **25%** to **400%**.

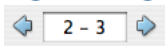
### Fit window

Scales the view of the active document so it completely fits into the Editor window.

### Fit width

Scales the view of the active document so it fits the width of the Editor window.

### Page navigation



This icon allows navigation through a multi-page document. Clicking on one of the arrow icons displays the previous or next page. The current page and the total number of pages are listed between the arrow icons.

It is also possible to enter the page number you want to navigate to directly into the display field.

### Cut

The selected elements are copied to the clipboard and removed from the workspace. Form elements retain all their properties (except tab order).



## Copy

The selected elements are copied to the clipboard. Form elements retain all their properties (except tab order).

## Paste

The contents of the clipboard are pasted onto the workspace (as far as this is possible). All properties retained when placing the elements into the clipboard are applied. This means that the position of the elements are the same as they were when the elements were copied.

## Delete

The selected elements are deleted from the workspace without being placed into the clipboard.

## Undo

The most recent action is undone

## Redo

The last action is repeated (also undoes the **Undo** command)

The next three icons are drop-down icons which work like submenus. The most recently used option remains displayed as an icon and can be directly repeated.

Drop-down icon

## Align

This drop-down icon opens the same submenu as the **Format —> Align** menu (see Fig. 3-14). The selected form elements are mutually aligned. The alignment happens according to the bounding box of the respective form elements.

The upper three items control the horizontal alignment, and the lower three items control the vertical alignment.

## Left

The elements are aligned horizontally to their left borders.

## Center

The elements are aligned horizontally to their centers.

## Right

The elements are aligned horizontally to their right borders.

## Top

The elements are aligned vertically to their top borders.

## Middle

The elements are aligned vertically to their centers.

## Bottom

The elements are aligned vertically to their bottom borders.

## Drop-down icon Make same size

This drop-down menu opens the same submenu as the **Format** —> **Make same size** submenu (see Fig. 3-15). The dimensions of the selected form elements are adjusted. The adjustments are made to the bounding box of the respective form elements.

## Same width

All selected elements get the same width as the reference element.

## Same height

All selected elements get the same height as the reference element.

## Same size

All selected elements get the same width and height as the reference element.

## Drop-down icon Spacing

This drop-down icon opens the same submenu as the **Format** —> **Spacing** menu (see Fig. 3-16). The distance between the selected elements are adjusted. The adjustment occurs on the vertical or horizontal center axis of the bounding box of the respective element. In order for this button to be available, at least three elements must be selected.

## Equal vertical spacing

The vertical spacing between the horizontal center lines of the selected elements are changed to be equal.

## Equal horizontal spacing

The horizontal spacing between the vertical center lines of the selected elements are changed to be equal.

## Function editor

Opens the Function editor. The Function editor is explained in section 3.2.8 and section 4.5.1.2.

### Stylesheet editor

Opens the Stylesheet editor. The Stylesheet editor is explained in section 4.7.1.

### Check form

Checks the form for inconsistencies and incorrect expression syntax, and generate an error report (see section 4.6).

## 3.2.4

### Controls bar

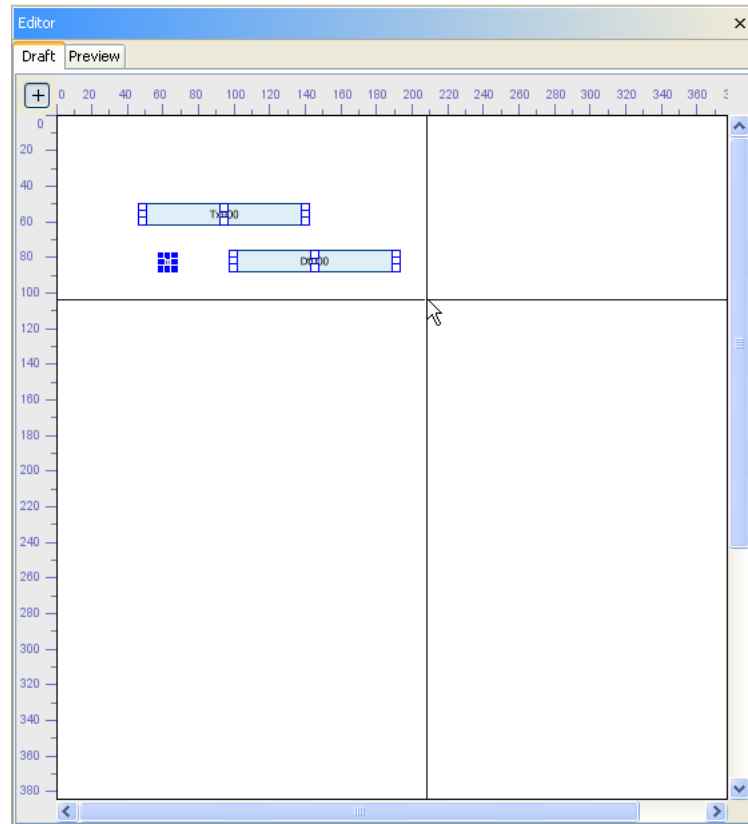
The Controls bar has the same structure as the **Controls** menu. For the elements of the Controls bar, refer to section 3.2.2.6.

## 3.2.5

### Workspace/Editor window


The workspace (see Fig. 3-22), also called **Editor** window, is the area in which forms are created and edited. The workspace corresponds to the paper format specified when a page is created.

**Fig.3-22** *The Editor window*



The Editor window can be switched between the Draft mode and the Preview mode (see section 3.3).

Starting from the origin in the upper left corner, there are scales with **Points** as measuring unit.

Clicking on the crosshair icon  shows or hides crosshairs, which can be very useful for positioning elements, as well as measuring positions on an existing background image. In Fig. 3-22, the crosshair is active.

The Close icon in the right upper corner has the same function as the menu item **File —> Close**.

## 3.2.6

## Properties window

Depending on the active control and the current selection, the Properties window (see Fig. 3-23 for a representative example) displays different information.

**Fig.3-23** Representative example for a Properties window

Properties of Multiline text field	
Property	Value
Name	M1000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	48.0978
Top	31.7935
Width	96.00
Height	46.2857
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Number of lines	0
Center single line	<input type="checkbox"/>
Link to object	

The Properties window displays property name/property value pairs. Depending on the property, the value is entered in a text or a numeric field, activated with a check box, or selected from a drop-down list. With some properties, such as Value, a new, specific editing window opens.

## 3.2.7

## Tools window

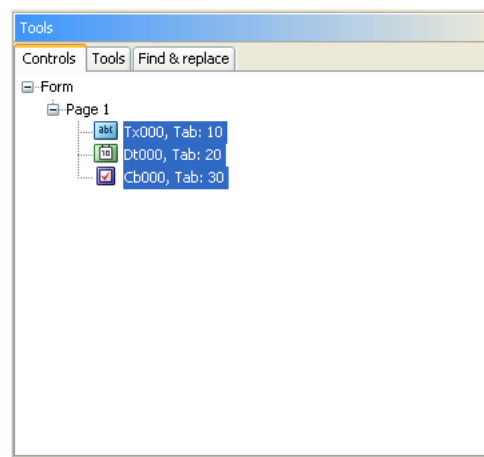
The **Tools** window is used to display additional information and tools. Clickable tabs display the object structure, a tool set or a Search/Replace window.

### 3.2.7.1

### Controls

When **Controls** is selected, the **Tools** window displays the object structure of the form (see Fig. 3-24). This is the default view of the Tools window.

**Fig.3-24** Object structure of the form



The object structure is hierarchical. When elements contain other elements, an opening icon appears. Clicking on this icon opens and closes the content of the element, providing more clarity in the display.

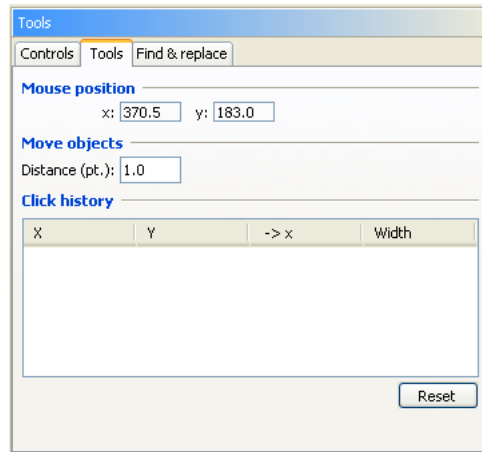
It is possible to select a single element or a group of elements. The Properties window is updated accordingly, and properties common for all selected elements can be set.

### 3.2.7.2

### Tools

The **Tools** window contains various settings which can make working with forms easier (see Fig. 3-25).

**Fig. 3-25** *The Tools window*



In the **Mouse position** area, the current position of the mouse cursor is continuously updated and displayed.

In the **Move objects** area, the distance can be specified (in **points**), by which objects are moved using the cursor keys. When the Shift key is pressed at the same time, the objects are moved 10 times the indicated distance.

When clicking on the workspace with the Control key pressed, the mouse position is entered into the **Click history** table. With every additional entry, the distance from the first point in the X direction is entered in the **-> x** column, and the distance in X direction from the previous point is entered in the **Width** column.

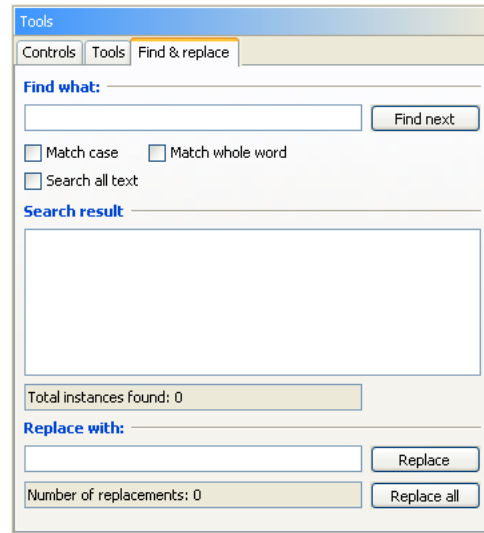
Clicking on the **Reset** button deletes the **Click history** table.

### 3.2.7.3

### Find and Replace

Selecting the **Find & Replace** tab shows the **Find/Replace** window (see Fig. 3-26).

**Fig. 3-26** *The Find / Replace window*



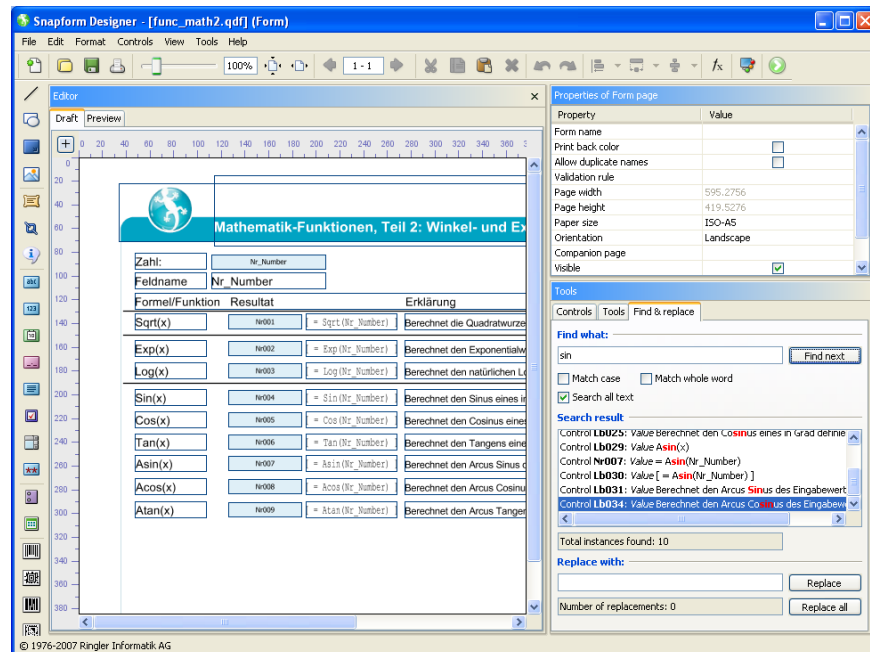
The entry field **Find what** holds a string of data. Clicking on the **Find next** button starts the search.

Normally strings are searched in the **Name**, **Value** and **Tooltip Text** properties. When the **Search all Text** box is checked, all fields and their values are searched.

By checking the **Match case** and **Match whole word** check boxes, the search is more precise. In the first case, upper and lower case letters are taken into consideration, and in the second case, the search string must be surrounded by “word boundaries” to create a match. When the original search string is not conclusive, these options can be very helpful.

The matches are displayed line by line in the **Search result** field (see Fig. 3-27).

**Fig. 3-27** Search results in the Find / Replace window



In this example, the string **sin** was searched for. This string appears in various fields. In the overview, the search string is highlighted in red. Double clicking the match activates the according element, which can then be edited.

In the **Replace with** field the replacement string is entered, which will replace the matches. Clicking on the **Replace** or **Replace all** buttons replaces the matches accordingly.

**Note:** When attempting to replace strings which may be part of expressions or are expression functions, extreme caution is required when clicking on "Replace all", in order to prevent faulty expressions.

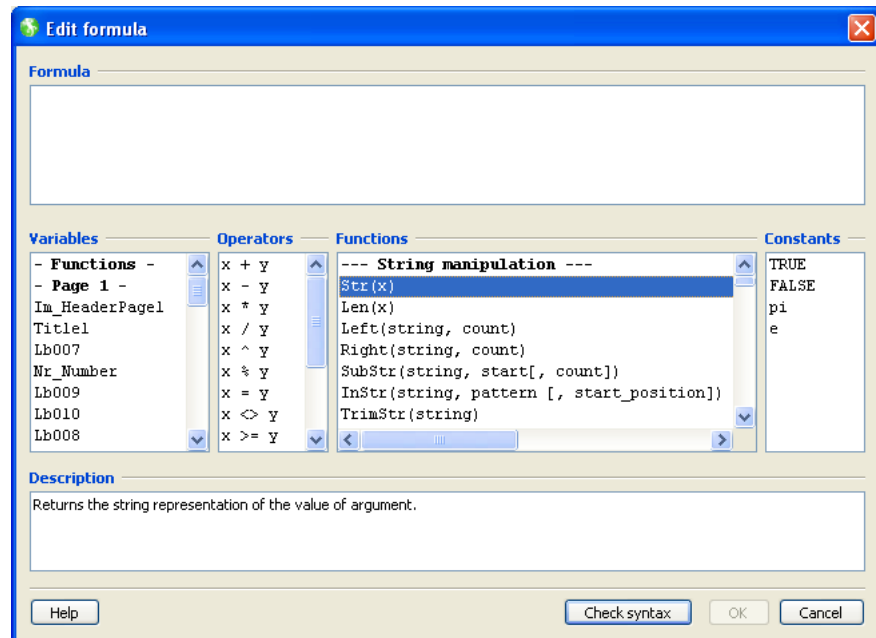
### 3.2.8

## Expression editor

When building smart electronic forms, Expressions (also called Formulas) are among the most important components. Expressions are assembled in the Expression editor (see Fig. 3-28). Functions are assembled in the identically looking Function editor.



**Fig. 3-28** *The Expression editor*



The **Formula** entry field is the actual workspace where an expression is entered.

An expression begins with an equal sign (=). When the equal sign is missing, the entry is treated as literal text and it will be shown as such.

The expression can either be entered directly, or it can be assembled from building blocks from the **Variables**, **Operators** and **Functions** fields. Double clicking on the building blocks transfers it to the **Formula** field.

The building blocks contain placeholders for arguments. These placeholders can be selected by double-clicking. Double-clicking on a building block to be entered replaces the placeholder with the selected entry.

The **Description** field displays a short descriptive text similar to the descriptions in section 5.3 of this manual.

Using this procedure, rather complex expressions can be assembled with a minimum of typing. When using functions including optional

arguments, care must be given to remove the brackets ([, ]), marking these arguments.

When the expression is assembled, it can be passed to the form by clicking **OK**. It is, however, recommended to verify the expression for formal correctness, by clicking on **Check syntax**. When the syntax checker finds an error, a message is displayed which displays indicators for the error. While it is possible to accept the erroneous expression, it must be corrected at a later time.

The expressions can be tested in the Preview (see section 3.3.2). However, in order to be sure that the expressions work correctly, they must be tested in the Snapform Viewer.

## 3.2.9 Other user interface components

The Snapform Designer has other user interface elements which are used for various purposes.

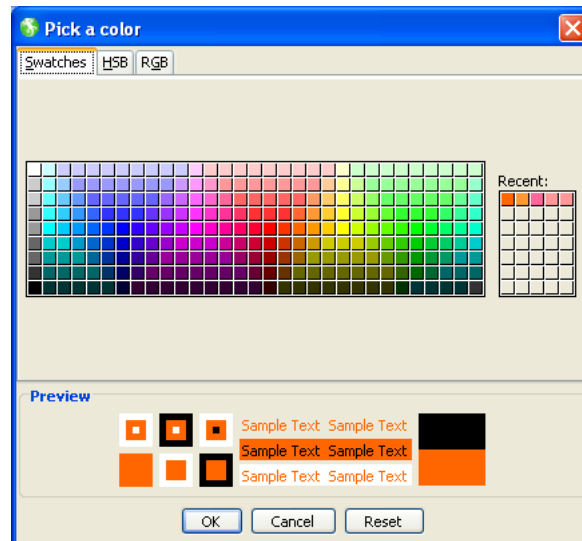
### 3.2.9.1 Keyboard Shortcuts

For experienced users, keyboard shortcuts are an efficient way to speed up the work. The Snapform Designer supports keyboard shortcuts. The key combinations are indicated in their appropriate menu item.

### 3.2.9.2 Color picker

Some field properties consist of color. In order to select a color, the Snapform Designer has a color selection mechanism which uses the color selection window (see Fig. 3-29). To select a color, three different color models are available. In the **Swatches** view(see Fig. 3-29) the color is selected from a palette of so-called “web safe” colors. In the **HSB** view (see Fig. 3-30), the HSB color model is used to select and define a color, and in the **RGB** view (see Fig. 3-31) the RGB values of the colors are specified.

**Fig. 3-29** Color picking window with Swatches

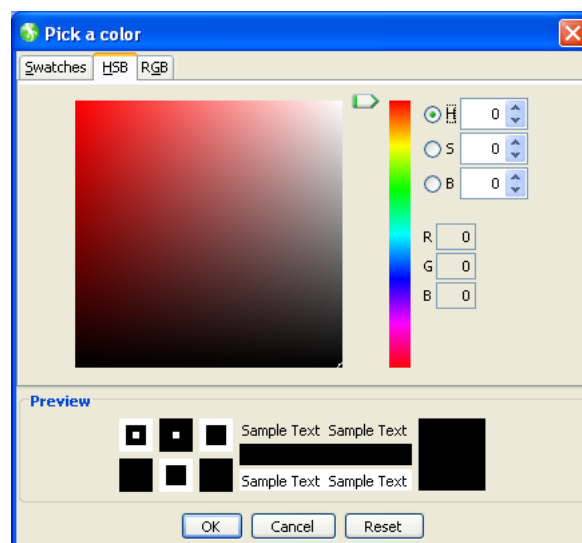


The color picking window is opened by clicking on the color bar in the Properties window. In the color selection mode using **Swatches** the color is selected by clicking on the according swatch. For each of these fields, the mouse tip displays the RGB values for the associated color.

In the **Preview** zone, the effect of the color selection is shown, as border and background of a field against field fill color, and against borders in white and black, as well as text color on a grey and white background, and as a background for black text.

The selected color is activated by clicking on **OK**.

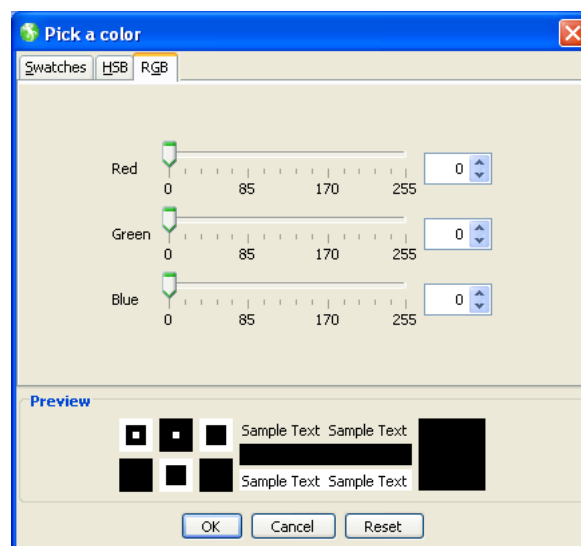
**Fig. 3-30** Color picker in HSB view



When selecting the color in the **HSB** view, the hue of the color is set using the vertical color spectrum slider in the middle of the window. The saturation can be set by clicking the appropriate place in the resulting colored square at the left. The current HSB and RGB values are displayed in the fields at the right.

HSB values can also be entered directly into the entry fields **H**, **S**, **B**, or increased or decreased using the **Up-Down** arrow icons. Values for **H** are between **0** and **360**, and for **S** and **B** they can be between **0** and **100**.

**Fig.3-31** Color picker in HSB view



In the **RGB** view, the colors are set using the three sliders for Red, Green and Blue. At the same time, the text fields at the right of the sliders show the current values.

It is also possible to enter the RGB values directly into the fields, or increase or decrease them with the **Up-Down** arrow symbols. The values for **R**, **G** and **B** can be between **0** and **255**.

## 3.3 Operation modes of The Snapform Designer

The Snapform Designer has two distinctive operation modes. The Draft view is the actual workspace for forms development. The Preview view represents the form as it would appear in the Snapform Viewer. Certain components of the form are displayed correctly only in Preview (such as HTML formatted text). The operation mode is switched by clicking on the according button in the title bar of the editor.

### 3.3.1 Draft mode

Draft mode is the actual workspace for forms development. In Draft view, elements can be placed on the workspace, and their properties can be changed.

In Draft mode, the tool bars and Controls bar are active. Also, all the menu items (if not limited for other reasons) are active.

### 3.3.2 Preview mode

Preview mode is used for verifying the look of the form, as well as testing expressions and the form's behavior when data is entered.

When switching to the Draft mode, all entered values are dropped and the fields are reset.

# 4 Forms Development

## 4.1 Introduction

Forms development is the main objective of the Snapform Designer. This chapter describes the various stages of the forms development process (analysis, base layout, active elements, logic, securing, deployment and management) using the Snapform Designer.

This chapter is process-based. It covers the sequences for creating working eForms. This uses several elements of the Designer. The scripting language for creating actions and logic is covered in detail in chapter 5, "Expression language reference".

The forms development process consists of the following stages:

1. Analysis and planning
2. Creating the base layout
3. Adding the data entry and display fields and the active elements
4. Adding actions and business logic
5. Preparing for deployment
6. Deployment
7. Maintenance

These steps are covered in this chapter in their own sections.

## 4.2 Analysis and planning

As in any development projects, analysis and planning have a great impact on the success of the project. Even if this stage is not directly connected with the use of the Snapform Designer, it is so important that a few comments are necessary.

As part of the analysis, the following questions must be asked (and answered):

- Is the business process associated with this form already fully known and understood?
- Is the form the only carrier of the associated business process?
- Which information is needed (input)?
- When (in the process) is a particular piece of information needed?
- What information must be displayed (output)?
- When in the process must a particular piece of information be displayed?
- Which output medium (screen, office printer, copier, printshop) is intended for the form?
- Which interactive elements are needed?
- Which plausibility tests and validations are needed?
- What kind of further calculations are needed?
- Will the form be pre-filled?
- Will the entered data be transferred to a back-end system, and if so, in which form?
- Are there any layout templates or style guides?
- Must the form be “accessible”?

- Should the form be filled out by the same user for different instances?
- Should the form be used exclusively in electronic workflows, or should it be possible to print out the empty form and fill it out by hand?
- Are there any repetitive elements?

Based on the answers to these questions, the form can now be defined and designed.

As part of the planning of the form, certain “highlight” colors must be defined. Snapform does this in the stylesheets, which are explained in section 4.7.

**Note:** *Stylesheets must be carefully planned. With well thought-out stylesheets, the work for further forms can be considerably reduced and they also ensure a common look and feel.*



## 4.3 Base layout

The first step of the development of a new form is the creation of the base layout. The base layout can either be imported as a background image (and would be non-editable in the Snapform Designer), or it can be part of the form (and be fully editable in the Snapform Designer). Mixed variants are of course always possible.

In many cases, a print version of the form already exists, very often in the PDF format. In such a case, the PDF document can be converted into a background image. This method has the advantage that the Snapform form will look identical to the print version (including micro-typographical finesses), and already existing material does not need to be recreated.

If there is no existing version, the base layout can be developed in the Snapform Designer, beginning with an empty page.

### 4.3.1 Background images

For background images, the **.hdt** format is used. Background images are created by converting a PDF document. For this conversions, two possibilities are available:

1. The Snapform PDF Converter
2. The free conversion service provided by Ringler Informatik AG

#### 4.3.1.1 Snapform PDF Converter

The Snapform PDF Converter is explained in the following section.

#### 4.3.1.2 Conversion Service

The Conversion Service provided by Ringler Informatik AG is available free of charge to all registered users of the Snapform Designer. PDF

documents are sent to Ringler Informatik AG, and the converted documents will be returned normally within 1 business day.

**Note:** *The original documents sent to the Conversion Service are deleted at Ringler Informatik AG after a successful conversion. The converted documents are deleted after a certain period (usually 10 business days). These documents are not made available to third parties in any way.*

## 4.3.2

### The Snapform PDF Converter

The Snapform PDF Converter is an application on its own which converts PDF documents into the Snapform background format (**.hdt**). The Snapform PDF Converter can convert a single document, or all documents in a given folder by using its batch processing capabilities.

The Snapform PDF Converter is a Windows application, based on the .NET framework. Its installation is explained in section 2.3.3.

#### 4.3.2.1

#### Preparing the PDF documents

The Snapform PDF Converter can process documents of any PDF version up to 1.6 (corresponding to Acrobat 7-compatible). Most documents can be processed without any further preparation.

However, it does make sense to prepare the PDF documents before running the conversion.

The preparation will occur in Acrobat Professional (Acrobat 6 or newer) or Acrobat full version (Acrobat 5 and older). The following instructions are based on Acrobat 7 Professional.

In the first step, any security setting must be removed.

The further preparation is done with the PDF Optimizer (menu **Advanced -> PDF Optimizer...**). The following settings are recommended:

1. Lower the resolution of images to 150 dpi (color and grayscale) or 600 dpi (b/w).
2. No further optimizing of scanned pages.
3. Embed fonts as far as possible (note that the font files must be installed on the computer used for the conversion).
4. Do not flatten transparent objects.
5. In the **Remove Objects** tab: select all options.
6. In the **Clean up** tab: Remove object compression and select all options except Optimizing for fast web view.

These options ensure that the document is prepared optimally for the Snapform PDF Converter.

**Note:** *With dynamic forms, make sure that all required form elements are displayed, and then flatten the document. In order to accomplish this, open Acrobat's Debugger (<Ctrl><J> or <Cmd><J>) and enter the following line of code:*

```
this.flattenPages() ;
```

*and execute it with <Ctrl><Return>.*

If there are several documents to be converted, the prepared PDF files can be copied to a specific directory. This will then allow the Snapform PDF Converter to operate in batch mode.

Another way to prepare the document (particularly if no Acrobat Professional version is available, but only Acrobat Standard or Acrobat Elements), is to "print out" again with Acrobat Distiller. This procedure, often called "refrying" is normally not recommended but for this purpose, it is a good method to simplify the document. For this conversion, the "additional" capabilities of PDF (links, fields, structure) are not necessary anyway.

In the Distiller settings make sure that the fonts will be fully embedded (also check these settings in the options for the printer driver).

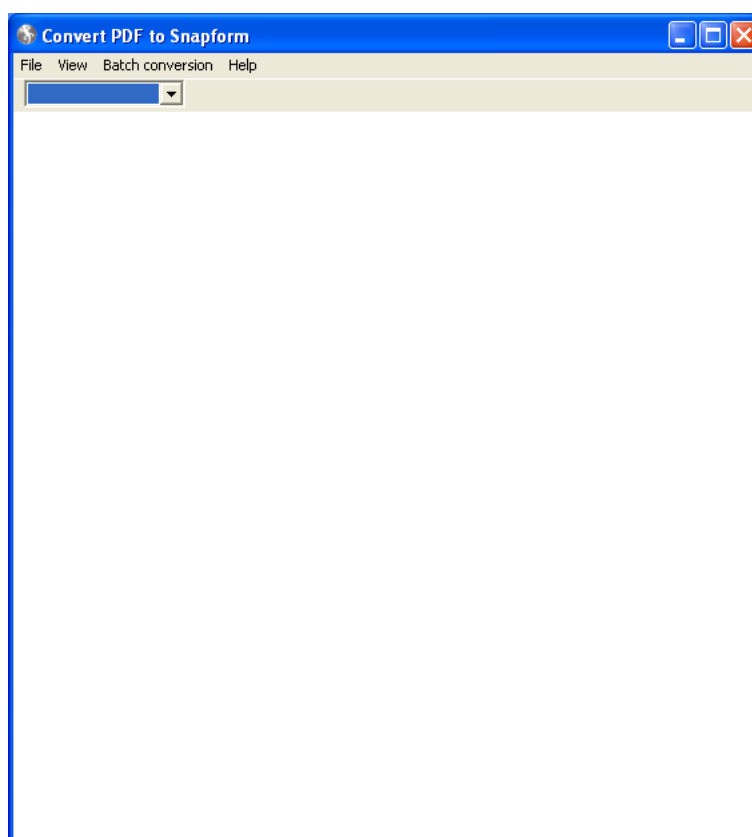
#### 4.3.2.2

### The user interface of the Snapform PDF Converter

The Snapform PDF Converter is only available with a user interface in English.

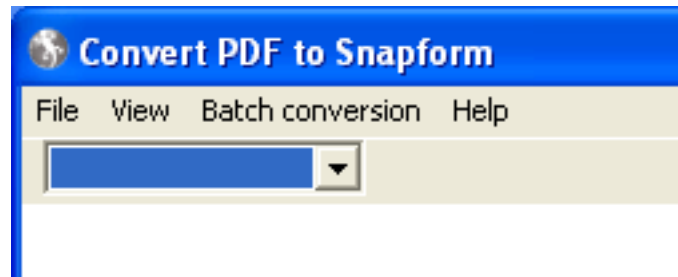
After starting up, the Snapform PDF Converter appears as shown in Fig. 4-1.

**Fig.4-1** *The Snapform PDF Converter*



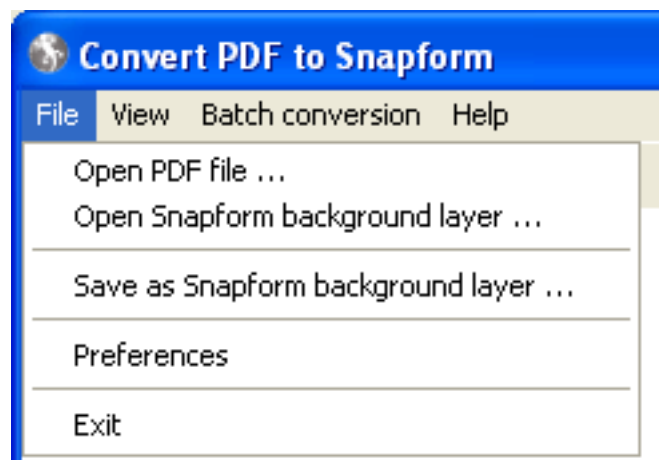
The menu bar is shown in detail in Fig. 4-2.

**Fig. 4-2** *The Snapform PDF Converter menu bar*



The various menu elements are explained below.

**Fig. 4-3** *The **File** menu of the Snapform PDF Converter*



The **File** menu (see Fig. 4-3) contains the following items:

### Open PDF file ...

A normal "File Open" dialog opens to select a PDF file. The selected file is converted and the result is displayed in the Preview window.

### Open Snapform background layer ...

A normal "File Open" dialog to select a Snapform background image file (.hdt) opens. The selected file is displayed in the Preview window.

### Save as Snapform background layer ...

A normal "File Save" dialog opens for selecting the location to save the current Snapform Background image. It will be saved in the **.hdt** format.

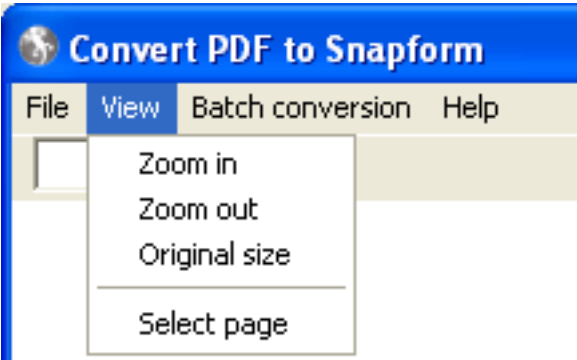
### Preferences

A dialog with some settings to optimize the conversion process opens.

Exit

The application exits.

**Fig.4-4** The **View** menu of the Snapform PDF Converter



The **View** menu (see Fig. 4-4) is used to control the display in the Preview window. It contains the following items:

Zoom in

Clicking in the Preview window enlarges the view by about 10%.

Zoom out

Clicking in the Preview window reduces the view by about 10%.

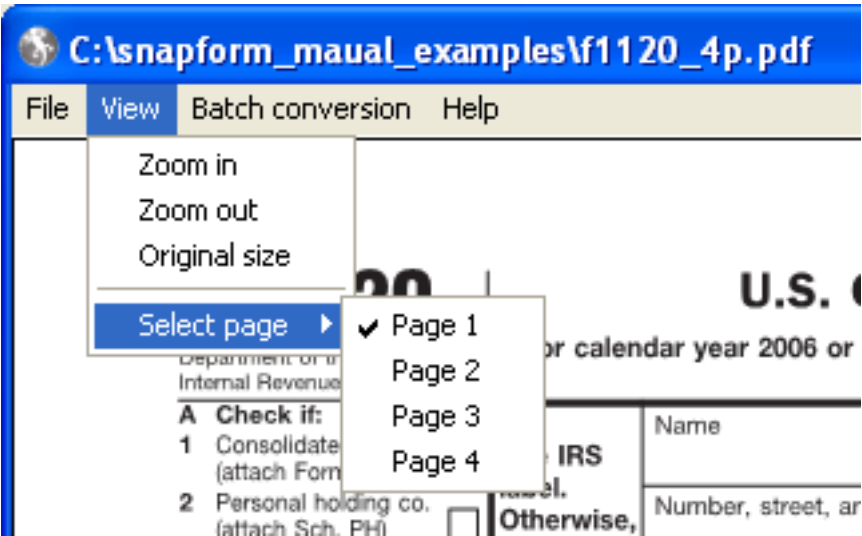
Original size

The view in the Preview window is displayed in original size (100%).

Select page

When a multiple page document is open, a submenu appears to select the pages (see Fig. 4-5). The current page is checked.

**Fig.4-5** The **Select page** submenu of the Snapform PDF Converter

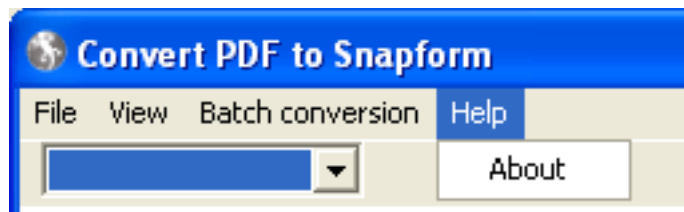


**Fig.4-6** The **Batch conversion** menu of the Snapform PDF Converter



The **Batch conversion** menu (see Fig. 4-6) contains only menu item **Convert PDF files...** which opens the dialog to select the folder whose files are to be converted in batch mode (see also section 4.3.2.5).

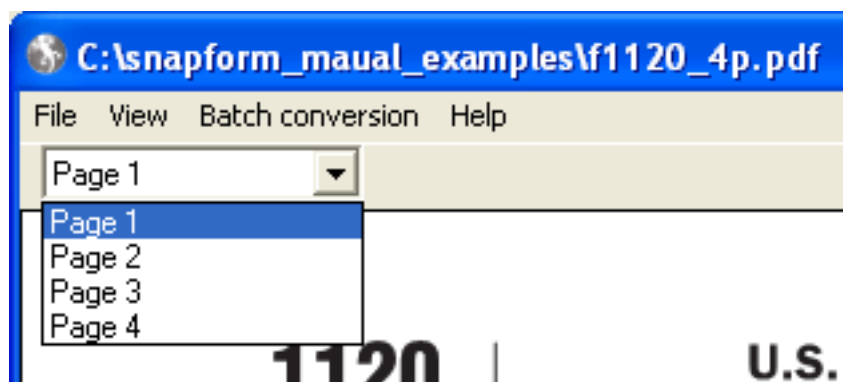
**Fig.4-7** The **Help** menu of the Snapform PDF Converter



The **Help** menu (see Fig. 4-7) contains as only menu item **About**, which opens a window with the program identification.

In addition to the **Select Page** submenu, there is a toolbar below the menu bar containing a drop-down menu with a page selector (see Fig. 4-8).

**Fig.4-8** Page selection drop-down menu

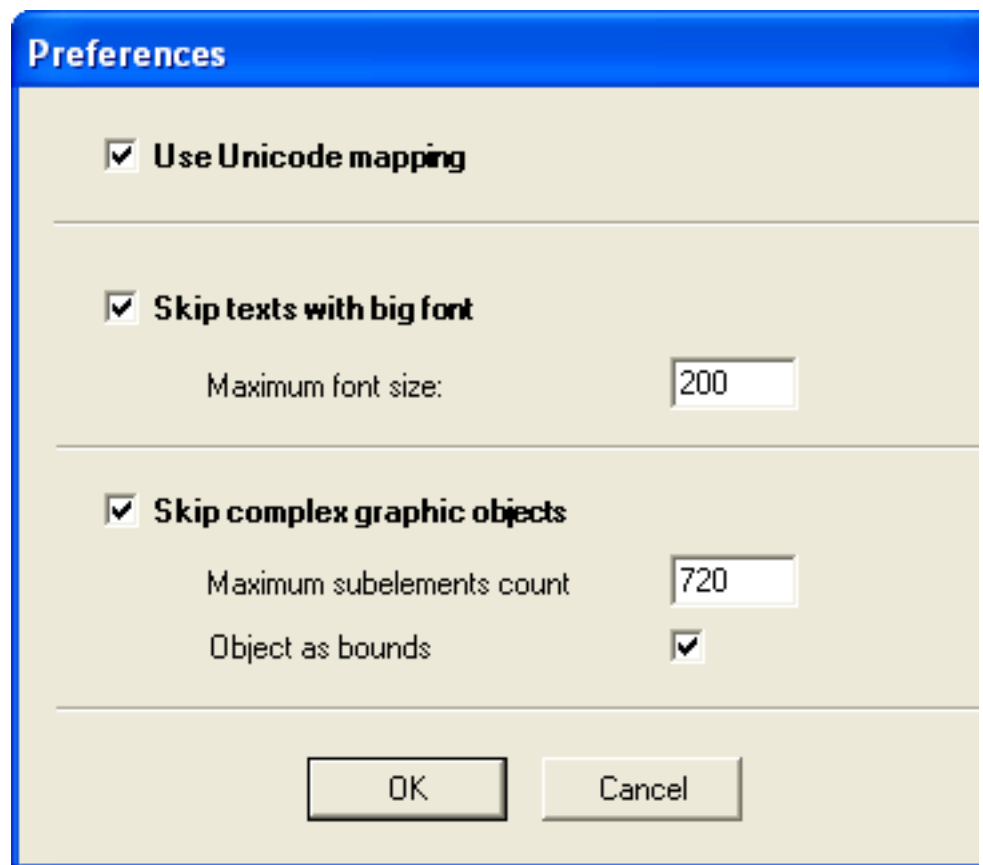


### 4.3.2.3

## Preferences

The conversion of a PDF document to a Snapform background image runs normally without problems, particularly when the PDF document has been prepared as described. There are, however, certain cases where the automatic conversion logic hits limitations, or the conversion itself takes an excessive amount of time. In such cases, certain parameters can be adjusted in the Menu **File -> Preferences** (see Fig. 4-9).

**Fig.4-9** *The Preferences dialog of the Snapform PDF Converter*



## Use Unicode mapping

Normally, a font is embedded and used in a PDF document with a defined character mapping table. There are cases when this table is missing or faulty, or when the version of the font installed on the local computer does not exactly match the font defined in the PDF document. Selecting this option forces the use of the character set encoding according to Unicode.



### Skip texts with big font

It is possible that the font size is very big (such as in a watermark), where it may be preferable to not convert such text.

This parameter suppresses the conversion of text with a font size greater than specified in the **Maximum font size** value. The default value is 200 Points.

### Skip complex graphic objects

Depending on the original document and the way the PDF has been created, there may be very complex vector paths. Converting such paths may involve considerable computing power, and increase the resulting file size.

This parameter suppresses the conversion of complex graphic elements where the number of nodes is beyond the entered threshold value.

The threshold value's (**Maximum subelements count**) default is 750 nodes.

By selecting **Objects as bounds** it is also possible to suppress the bounding path of the graphic object. When this option is selected, the graphic object will not be converted in any way.

#### 4.3.2.4

### Converting a single document

After having been prepared, the PDF document can be opened in the Snapform PDF Creator with the menu item **File -> Open PDF file....** After a few moments, the converted document appears in the Preview window, where it can be checked and then saved.

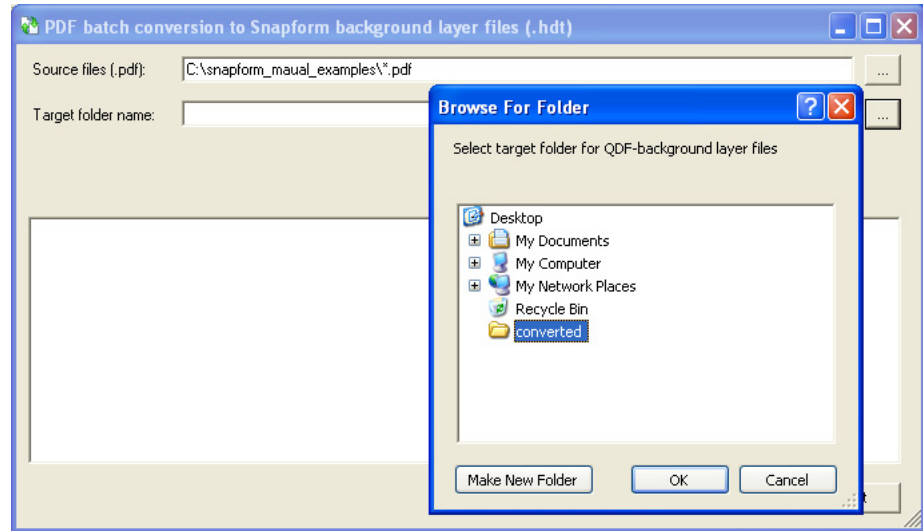
#### 4.3.2.5

### Batch conversion

If there are several PDF documents to be converted, it is recommended to use the batch conversion feature.

The prepared documents to be converted are collected in a specific directory. **Batch conversion -> Convert PDF files...** opens the batch conversion dialog (see Fig. 4-10).

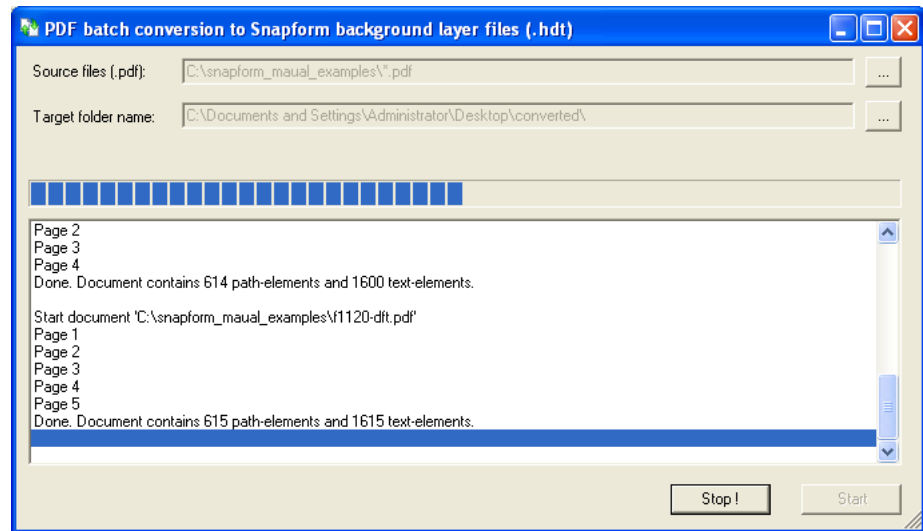
**Fig. 4-10** *Batch conversion window with open Select directory dialog*



In the first line **Source files (.pdf):** the path of the source directory containing the PDF documents is specified. Clicking the ... button opens a Select Directory dialog to specify the according directory. In the same way, the target directory is specified in the **Target folder name:** line.

When both directories are selected, start the conversion by clicking on the **Start** button. All PDF documents found in the source directory will now be converted one after the other. The progress is indicated with a progress bar. The conversion for each document is also logged in the log window. In addition to an entry for the document, there is an entry for each page of the document, and the number of converted path and text elements is shown (see Fig. 4-11).

**Fig. 4-11** *Progress of the batch conversion.*



Both the source and the target directory must be specified. If an entry is missing, the message **Source files are not specified** or **Target folder is not specified** appears after clicking on the **Start** button.

The batch conversion can be stopped any time by clicking the **Stop!** button.

When the **Close** button is visible, the process has completed. Clicking the **Close** button closes the batch conversion dialog.

#### 4.3.2.6

#### Conversion problems

A conversion may not complete properly. In such a case, the according PDF document should be opened in Acrobat Professional and then viewed page per page. Look for any error messages from Acrobat. In many cases, saving the document under another name will solve the issue. If the problem persists, try to print the document again, using the Acrobat Distiller/Adobe PDF printer (see section 4.3.2.1).

If the conversion always fails with a specific page, try to extract the page from the document and convert it separately. Also remove the page and convert the remaining document. These two parts can be joined again in the Snapform Designer using the **Add pages** function.

Because of differences in the internal data formats of the PDF and the Snapform Background image, it is possible that certain style elements are not be entirely converted. It is up to the forms designer to decide whether these elements are important (and will be prepared in the PDF document), or whether the result of the conversion is acceptable.

When running a batch conversion, it is possible that the converter gets stuck with a certain file, and clicking the **Stop!** button does not stop the conversion. In this case, close the window with the close box; this will reliably stop the conversion process. The file causing the problem must now be checked and prepared again for the conversion, or be removed from the source directory.

### 4.3.3

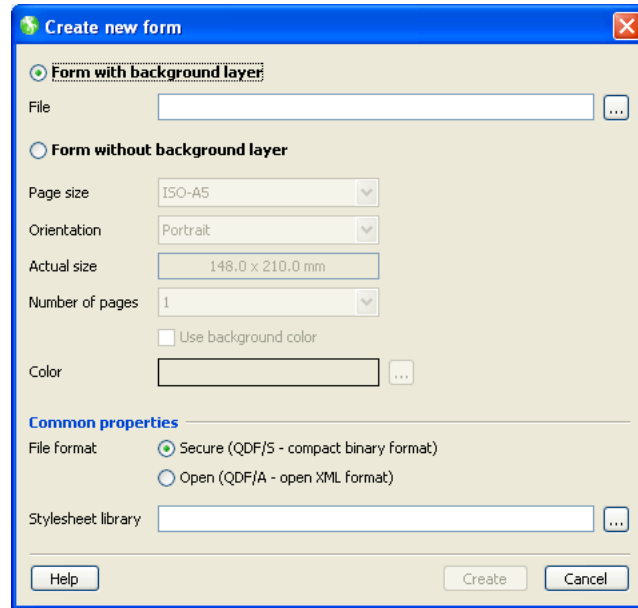
## Creating a new document

The conversion of a PDF document into a Background image is essentially a preliminary step to create a new QDF document, but it can be done independent of further actions.

The actual work with the Snapform Designer begins with the creation of a new document. New documents can be created either as empty pages, or by importing existing documents.

The menu item **File -> New Form...** opens the **Create new form** dialog (see Fig. 4-12).

**Fig. 4-12** *The Create new form dialog*



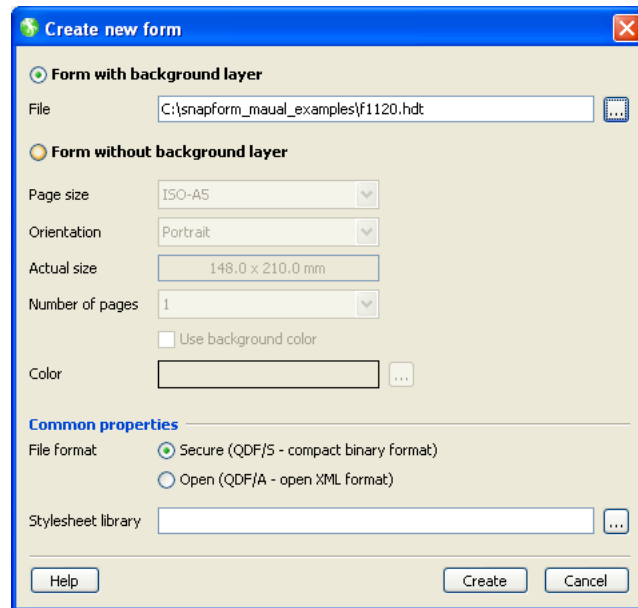
The Create new form dialog consists of three areas: **Form with background layer**, **Form without background layer** and **Common properties**. The last area is relevant for either other option.

#### 4.3.3.1

#### Importing existing background pages

The fastest way to a new form is when there is already an existing background image as a **.hdt** file. When the option **Form with background layer** is selected, an entry line for the path to the **.hdt** file becomes active. The **...** button opens a normal File Open dialog which allows selection of the file (see Fig. 4-13).

**Fig. 4-13** *New form with background layer.*



The resulting page size depends on the dimensions of the background image. The number of pages also depends on the number of pages in the background image document.

After selecting the **.hdt** file, several properties must be specified to create the form (see section 4.3.3.3).

**Note:** *In a form created in this way, the background image cannot be edited in the Snapform Designer. When the background image must be edited, it has to happen in the original document from which the PDF has been generated, and the background image has to be created again.*

### 4.3.3.2

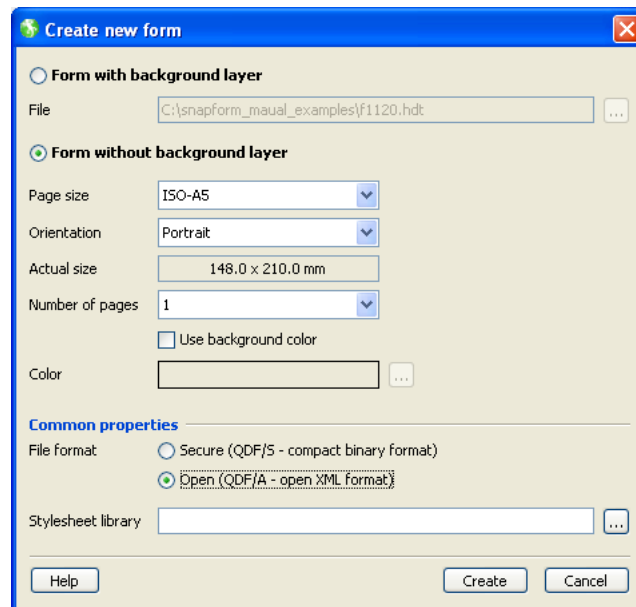
### Creating empty pages

If there is no suitable background image available in the **.hdt** format, or the chances that the base layout will be frequently changed in the future, it is recommended to start from a completely blank document.

**Note:** *It is always possible to add a background to an existing form (see section 4.3.4.2).*

When the option **Form without background image** is selected, the **Create new form** dialog has 5 parameters available (see Fig. 4-14).

**Fig.4-14** *New form without background layer*



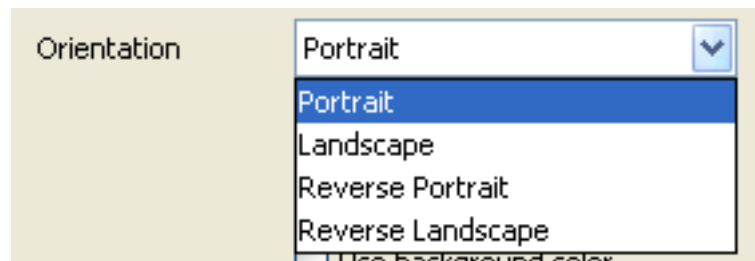
## Page size

This drop-down menu contains a comprehensive list of paper sizes from all over the world.

## Orientation

This drop-down menu contains a list of the available page orientations (see Fig. 4-15).

**Fig.4-15** *drop-down Menu Orientation*



The options **Reverse portrait** and **Reverse landscape** are intended for places where both sides of the paper are going to be printed.

## Actual size

Dimensions of the selected paper size. This field cannot be edited.

## Number of pages

Number of pages to create. The drop-down menu allows to select 1 to 20 pages.

If more pages are needed, they can be added easily (see also section 4.3.4.1).

## Use background color

When this box is checked, the background of the form will be in the color specified in the **Color** selection field.

## Color

When the check box **Use background color** is checked, this field becomes active. Double-clicking this field or clicking on the ... button opens the color picking window (see section 3.2.9.2), where the color can be specified.

### 4.3.3.3

## Properties of new forms

After selecting the background image or the parameters for the blank form, two more parameters are necessary before the form can be created. These parameters are part of the Common properties area (see Fig. 4-16).

**Fig.4-16** *Properties of the new form*

**Create new form**

☐ Form with background layer

File: C:\snapform\_manual\_examples\ff1120.hdt

☒ Form without background layer

Page size: ISO-A5

Orientation: Portrait

Actual size: 148.0 x 210.0 mm

Number of pages: 3

☒ Use background color

Color: [Blue swatch] [Color picker icon]

**Common properties**

File format: ☐ Secure (QDF/S - compact binary format) ☒ Open (QDF/A - open XML format)

Stylesheet library: [Text box] [Button]

[Help] [Create] [Cancel]



## File format

The Snapform file format exists in two varieties: encrypted/compressed and open.

When **Secure (QDF/S)** is selected, the file will be completely encrypted and compressed when it is saved. This leads to the smallest possible file size, but requires a Snapform application to read the data.

When **Open (QDF/A)** is selected, the payload of the file will be saved in the clear in XML form (layout and background layer will be compressed in any case). This leads to slightly bigger files. It is however possible to read the data without any Snapform applications (e.g. with a Document Management System).

The preferred variety depends on the planned workflow for the forms, and must be decided on a case-by-case level.

## Stylesheet library

It is possible to store field properties of fields and other form elements in a stylesheet. This stylesheet can be directly included when the form is created. With this, all properties defined in the stylesheet are automatically selected as default for any newly placed elements.

Stylesheets are explained in section 4.7.

Clicking on the ... button opens a File Open dialog which allows to select the according file.

With these options set, the new form is specified, and it can be created by clicking on the **Create** button.

### 4.3.4

## Document pages

Forms are page-oriented. Snapform Designer provides functions which allow you to manage pages in forms. Pages can be added and removed, and pages have a set of properties. In multiple page forms, each page may have its own unique page properties.

#### 4.3.4.1

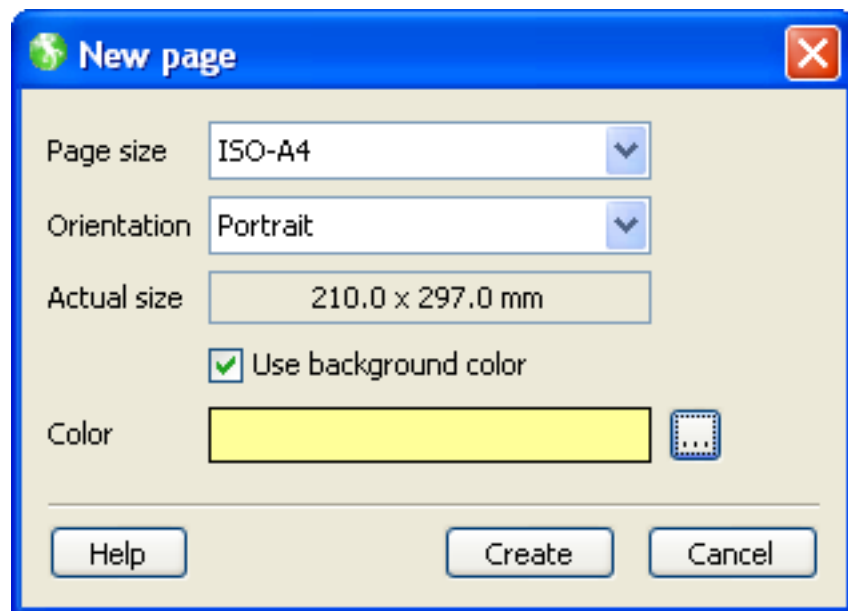
### Adding and removing pages

To add and remove pages, three items are available in the **Edit** menu.

#### Add new page

The menu item **Edit -> Add new page...** opens the **New page** dialog (see Fig. 4-17). This dialog is very similar to the one to create a new page without background image.

**Fig. 4-17** Add page dialog



#### Page size

This drop-down menu contains a comprehensive list of paper sizes from all over the world.

#### Orientation

This drop-down menu contains a list of the available page orientations (see Fig. 4-15).

The options **Reverse portrait** and **Reverse landscape** are intended for places where both sides of the paper are going to be printed.

#### Actual size

Dimensions of the selected paper size. This field cannot be edited.

#### Use background color

When this box is checked, the background of the form will be in the color specified in the **Color** selection field.

## Color

When the check box **Use background color** is checked, this field becomes active. Double-clicking this field or clicking on the ... button opens the color picking window (see section 3.2.9.2), where the color can be specified.

Clicking on **Create** creates the page and adds it to the document. The new page also becomes the current page.

## Insert new page

The menu item **Edit → Insert new page ...** is similar, but inserts the new page before the current page. The new page also becomes the current page.

## Deleting pages

Pages are removed with the menu item **Edit → Delete this page**. This command deletes the current page.

**Attention:** *This command deletes the current page without further warning and without the possibility of undoing.*

**Note:** *After the current page has been deleted, the first page of the document becomes the current page.*

### 4.3.4.2

## Page properties

The page properties are displayed in the Properties window (see Fig. 4-18), as long as no element of the page is selected. Page properties are normally valid for the current page, but they may also be valid for the whole form (see descriptions).

**Fig. 4-18** *The Page Properties window*

Properties of Form page	
Property	Value
Form name	
Print back color	<input type="checkbox"/>
Allow duplicate names	<input type="checkbox"/>
Validation rule	
Page width	419.5276
Page height	595.2756
Paper size	ISO-A5
Orientation	Portrait
Companion page	
Visible	<input checked="" type="checkbox"/>
Printable	<input checked="" type="checkbox"/>
Background image	

The first three properties are valid for the whole form, the others are limited to the current page.

### Form name

Allows you to apply a name to the form. This name is used as an internal reference.

### Print back color

When this option is checked, the background color is printed as well. When it is not checked, no background color is printed. This has the advantage that the visual requirements on screen, as well as the requirements for printing can be honored.

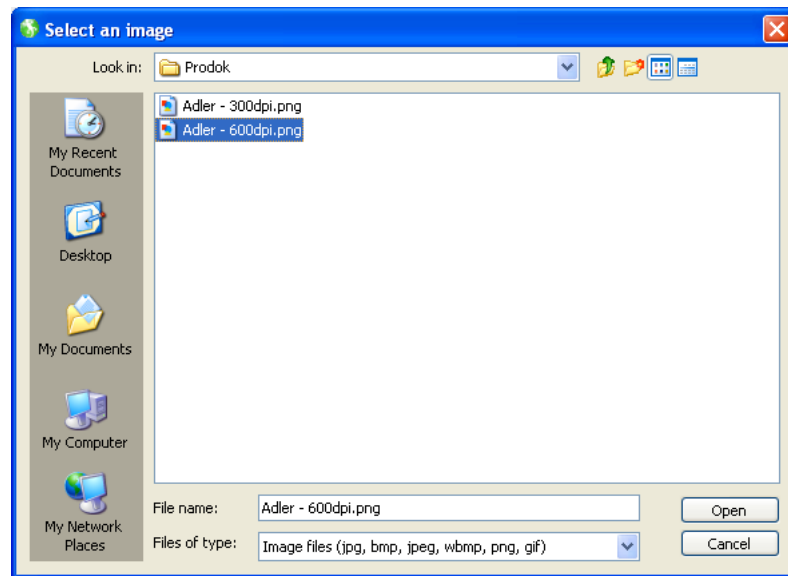
### Allow duplicate names

This option allows fields to have the same name. When this is the case, fields can be duplicated over all the pages of the form, and when filled in, they automatically get the same value.

<b>Validation rule</b>	The expression editor opens and an expression which will be executed when the page opens can be entered.
<b>Page width</b>	Width of the page in the default measuring unit. This field is for informational purposes only and cannot be edited.
<b>Page height</b>	Height of the page in the default measuring unit. This field is for informational purposes only and cannot be edited.
<b>Paper size</b>	Selecting this property opens a drop-down menu with a list of the paper sizes, as it is available when inserting pages or defining new documents. The selected paper size is applied immediately.
<b>Orientation</b>	Selecting this property opens a drop-down menu with a list of the page orientations, as it is available when inserting pages or defining new documents.
<b>Companion page</b>	When selecting this property, a drop-down menu opens with the list of the pages of the document. This property is used for the pagination of the finished document.
<b>Visible</b>	This check box controls whether the current page is visible or not.
<b>Printable</b>	This check box controls whether the current page can be printed or not.
<b>Background image</b>	<p>When selecting this property, the path to a graphic file can be specified which will be inserted as a background image. When there are already inserted background images, their names are listed in a drop-down menu.</p> <p>The topmost element of the drop-down menu is empty, and has the meaning of "no image". Selecting this element removes an already inserted background image from the page.</p>

The ... button opens a File Open dialog to select the file to be imported (see Fig. 4-19).

**Fig.4-19** *FileOpen dialog for background image*



Graphic files of the formats JPEG, BMP, TIFF, GIF and PNG can be imported as background images. Other graphic formats, in particular vector-based formats, must first be converted/rendered into one of the listed formats.


The imported background image is scaled to completely fill the page, which means that its proportions are not maintained.

**Helpful Hint:** *When the background image is printed out, be aware that the JPEG format may show artifacts in line art or text images. These artifacts will appear as spots in the printout. line art or text images should be preferably imported in the TIFF or PNG format.*

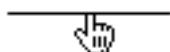
## 4.3.5 Graphical elements

Snapform has four graphical elements: lines, rectangles, (cover) areas and images. The graphical elements are considered to be part of the base layout. They are part of the form structure, and therefore independent from background images.

### 4.3.5.1 Lines

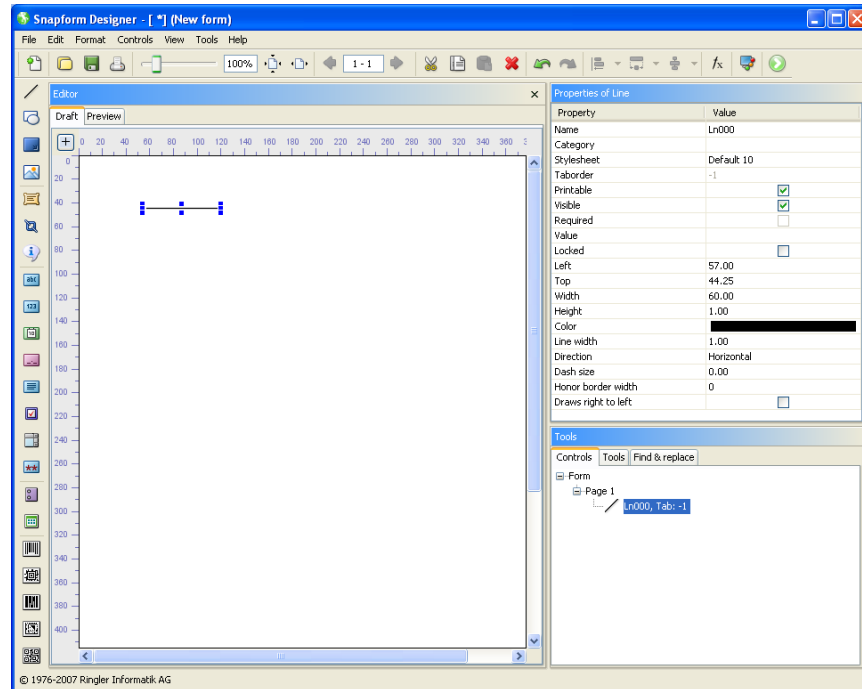
The Line tool  creates simple straight lines. After selecting the tool, a cursor with a horizontal line appears (see Fig. 4-20).

**Fig.4-20** *Line tool cursor*



Associated with the Line tool is a black horizontal line, 1 pt wide and 60 pt long. Clicking the mouse places it on the workspace. The line automatically gets a default name, consisting of the characters **Ln** and a consecutive three-digit number, beginning with **000**. The first line placed in a form has therefore the name **Ln000**. In the object structure, it is added to the according page. In Fig. 4-21 the first line of the form has been placed and then selected.

**Fig. 4-21** *Newly created line*




The length and the position of the line can be changed by dragging the anchor points of the bounding box. The position of the line can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow. Note that the direction of the line does not change even when the bounding box is increased perpendicular to the line; the line remains centered to the middle of the bounding box.

The other features of the line are controlled in the Properties window (see Fig. 4-22).



**Fig. 4-22** *Properties of a line*

Properties of Line	
Property	Value
Name	Ln000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	57.00
Top	44.25
Width	60.00
Height	1.00
Color	
Line width	1.00
Direction	Horizontal
Dash size	0.00
Honor border width	0
Draws right to left	<input type="checkbox"/>

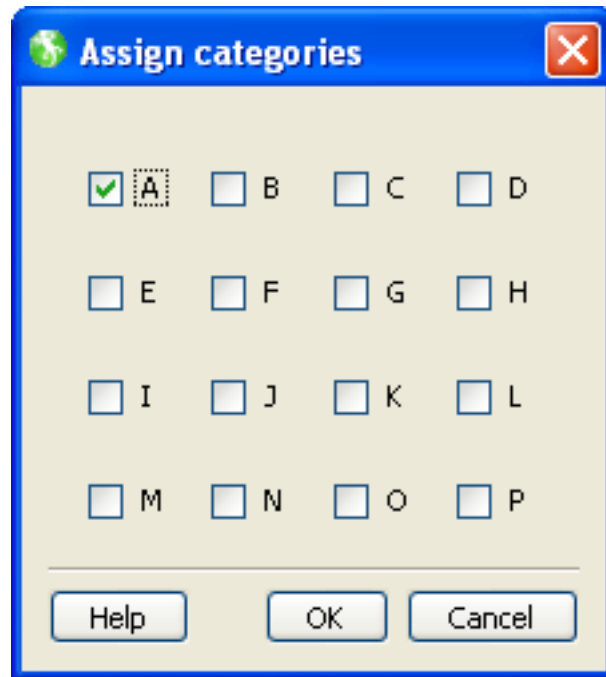
## Name

The name of the line in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

**Fig. 4-23** *The Assign categories dialog*



The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For lines, the stylesheet definition has however no relevance.

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the line is not an active element.

## Printable

When the box is checked, the line will be printed.

## Visible

When the box is checked, the line is displayed on screen.

## Required

This property is always deactivated, and cannot be changed.

## Value

Clicking on this field opens the expression editor for assembling an expression. For Line elements, an expression is however used very rarely.

## Locked

When this box is checked, the line gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog become inactive. Access to these properties is resumed only after deactivating this check box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the line.

**Note:** *For a precise positioning of the line's center half of the line's width must be taken into account.*

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the line.

**Note:** *For a precise positioning of the line's center half of the line's width must be taken into account.*

## Width

Width of the bounding box of the line. With a horizontal line this is the same as the length of the line; with a vertical line, it is the same as the line's width.

## Height

Height of the bounding box of the line. With a horizontal line this is the same as the width of the line; with a vertical line, it is the same as the line's length.

## Color

Color of the line. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

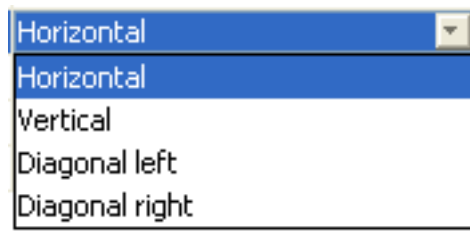
## Line width

Width of the line in Points.

## Direction

Selecting this property opens a drop-down menu with the possible directions of the line (see Fig. 4-24).

**Fig. 4-24** *Direction drop-down menu in Line properties*



The options have the following meaning:

### Horizontal

The line runs horizontally from left to right.

### Vertical

The line runs vertically from top to bottom.

### Diagonal left

The line runs diagonally in its bounding box from the bottom left corner to the top right corner.

### Diagonal right

The line runs diagonally in its bounding box from the top left corner to the bottom right corner.

## Dash size

This value determines whether the line is displayed whole or dashed. The value **0** means that the line is displayed whole. Other values indicate that the line is displayed dashed, where the value is the length of the dash.

**Helpful Hint:** *A dotted line can be simulated by setting the dash size to the line width.*

## Honor border width

The stylesheet defines an element border which is normally not relevant for lines. However, when placing a line below an entry field with a visible border, there can be a visible discrepancy between the line length and the field perimeter (because the field's border is rendered outside of the actual field definition rectangle. This property increases the length of the

line when it is rendered (in Preview mode or in the Snapform viewer) by the width of the border, as specified in the stylesheet, so that it appears to be of the same size as the entry field with its border.

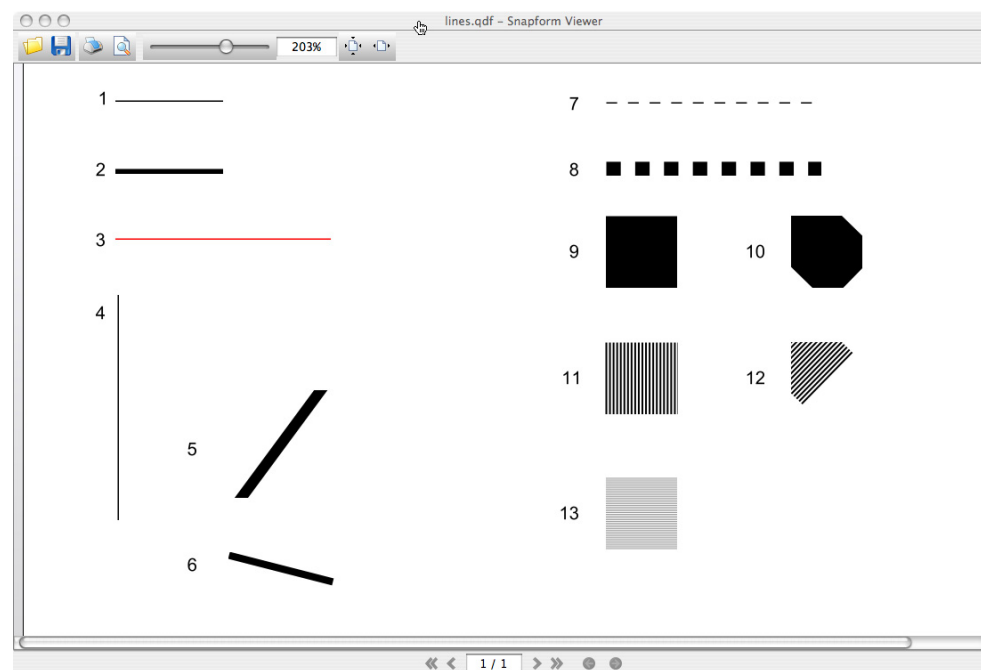
## Draws right to left

There are cases where the direction of the line matters, such as for dashed lines, and (in other contexts) for arrows. Normally, a line is drawn from left to right, but when this box is checked, it is drawn from right to left.

**Helpful Hint:** *It is most efficient for the workflow to place the line associated to the Line tool cursor into the vicinity of the intended position, and then completely define it via the properties window.*

The document *linien.qdf* (see Fig. 4-25) shows various examples of Lines. After selecting a line in the Snapform Designer, the properties window displays information about the features of that line.

**Fig.4-25** Line examples  
from *linien.qdf*




- 1 Ln000: Default Line: horizontal, 60 pt long, 1 pt wide, black
- 2 Ln001: horizontal, 60 pt long, 3 pt wide, black
- 3 Ln002: horizontal, 120 pt long, 1 pt wide, red
- 4 Ln003: vertical, 125 pt long, 1 pt wide, black

- 5 Ln004: diagonal left, bounding box 60 x 60 pt, 6 pt wide, black
- 6 Ln005: diagonal right, bounding box 60 x 20 pt, 4 pt wide, black
- 7 Ln006: horizontal, 120 pt long, 1 pt wide, dash size 6 pt, black
- 8 Ln007: horizontal, 120 pt long, 8 pt wide, dash size 8 pt, black
- 9 Ln008: horizontal, 40 pt long, 40 pt wide, black
- 10 Ln009: diagonal right, bounding box 40 x 40 pt, 40 pt wide, black
- 11 Ln010: horizontal, 40 pt long, 40 pt wide, dash size 1 pt, black
- 12 Ln011: diagonal right, bounding box 40 x 40 pt, 40 pt wide, dash size 1 pt, black
- 13 Ln012: vertical, 40 pt long, 40 pt wide, dash size 0.5 pt, grey

### 4.3.5.2

## Rectangles

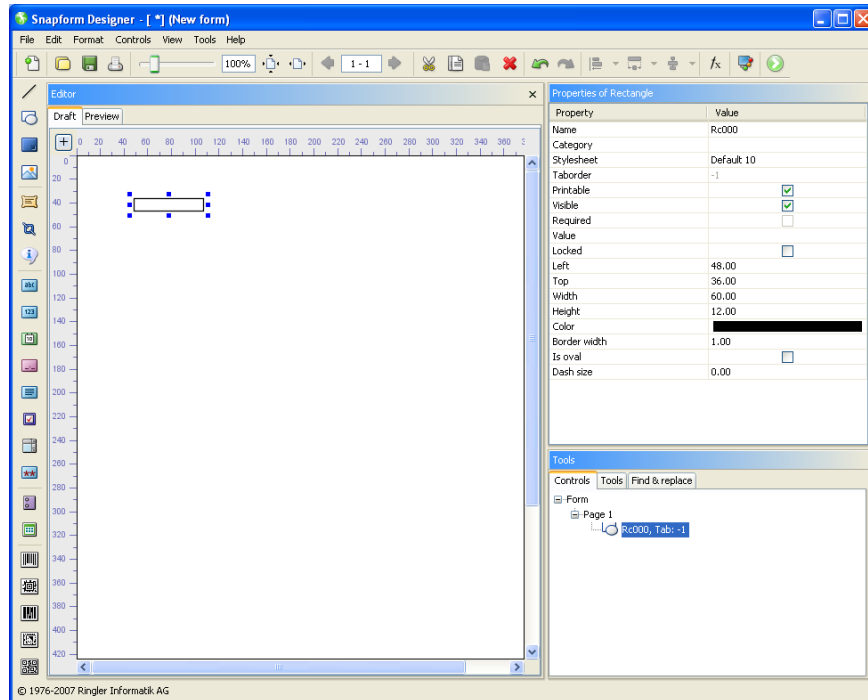
The Rectangle tool  creates rectangular and oval lines. After selecting the tool, a cursor with a rectangle in default size appears (see Fig. 4-26).

**Fig. 4-26** *Rectangle tool cursor*



Assigned to the Rectangle tool is a black rectangle, 60 pt wide and 12 pt high, with a line width of 1 pt. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Rc** and a three-digit consecutive number, beginning with **000**. The first rectangle placed in a form has therefore the name **Rc000**. In the object structure it is added to the according page. In Fig. 4-27 the first rectangle of the form has been placed and then selected.


**Fig. 4-27** *Newly created rectangle*



The length and the width of the rectangle can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the rectangle are controlled in the Properties window (see Fig. 4-28).

**Fig. 4-28** *Properties of a rectangle*

Properties of Rectangle	
Property	Value
Name	Rc000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	48.00
Top	36.00
Width	60.00
Height	12.00
Color	
Border width	1.00
Is oval	<input type="checkbox"/>
Dash size	0.00

## Name

The name of the rectangle in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For rectangles, the stylesheet definition has however no relevance.



<b>Taborder</b>	Sequence number of the element in the tab order. This value is <b>-1</b> and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the rectangle is not an active element.
<b>Printable</b>	When the box is checked, the rectangle will be printed.
<b>Visible</b>	When the box is checked, the rectangle is displayed on screen.
<b>Required</b>	This property is always deactivated, and cannot be changed.
<b>Value</b>	Clicking on this field opens the expression editor for assembling an expression. For Rectangle elements, an expression is however used very rarely.
<b>Locked</b>	When this box is checked, the rectangle gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog become inactive. Access to these properties is resumed only after deactivating this check box.
<b>Left</b>	X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the rectangle.
<b>Top</b>	Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the rectangle.
<b>Width</b>	Width of the bounding box of the Rectangle element.
<b>Height</b>	Height of the bounding box of the Rectangle element.

**Note:** *The dimensions of the rectangle are measured on its outside; if several rectangles are placed to form a table, the line width must be taken into account.*

## Color

Color of the rectangle. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

## Line width

Width of the border of the Rectangle element in Points.

The line width is measured from the bounding box inward.

**Helpful Hint:** *Rectangles which are narrower than their line width may not be shown correctly.*

## Is oval

When this box is checked, an ellipse, limited by the bounding box of the element is displayed instead of a rectangle.

## Dash size

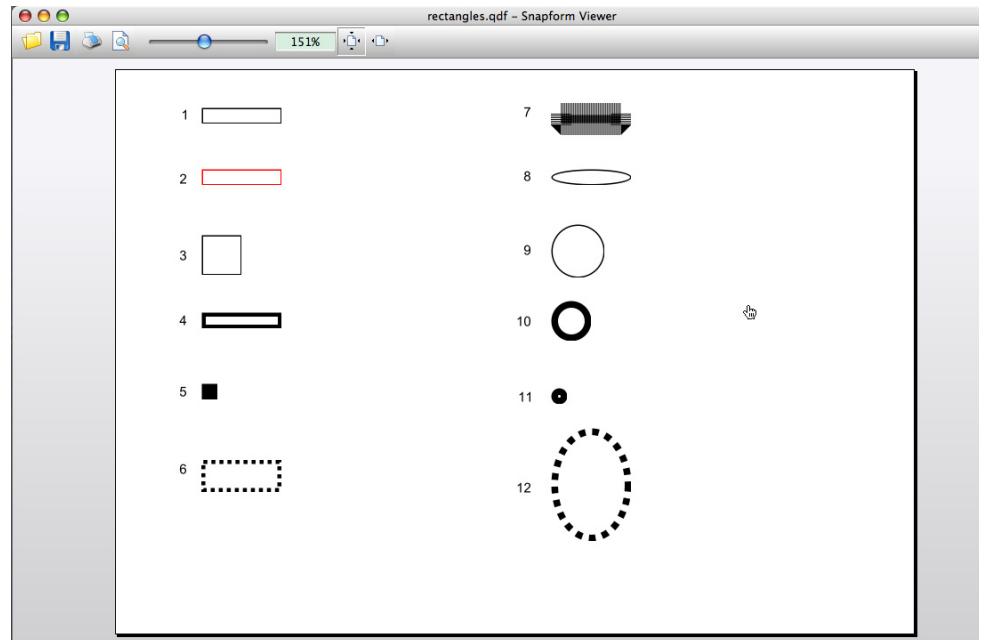
This value determines whether the rectangle is displayed whole or dashed. The value 0 means that the line is displayed whole. Other values indicate that the line is displayed dashed, where the value is the length of the dash.

With rectangles, the dashing starts at the upper left corner and runs clockwise. With ellipses, the dashing starts at the contact point with the right hand side of the bounding box (3-o'clock position) and runs clockwise.

The document *rectangles.qdf* (see Fig. 4-29) shows various examples of Rectangles. After selecting a rectangle in the Snapform Designer, the properties window displays information about the features of that rectangle.

**Note:** *The behavior of dashed lines can be verified best in Snapform Designer with this sample file, by selecting items 6 or 12, and then changing their size.*

**Fig. 4-29** Sample rectangles  
from  
*rectangles.qdf*




- 1 Rc000: Default rectangle, 60 pt wide, 12 pt high, border width 1 pt, black
- 2 Rc001: Rectangle, 60 pt wide, 12 pt high, border width 1 pt, red
- 3 Rc002: Square (rectangle), 40 pt wide, 40 pt high, border width 1 pt, black
- 4 Rc003: Rectangle, 60 pt wide, 12 pt high, border width 3 pt, black
- 5 Rc004: Square (rectangle), 12 pt wide, 12 pt high, border width 6 pt, black
- 6 Rc005: Rectangle, 60 pt wide, 24 pt high, border width 3 pt, black, dash size 3 pt
- 7 Rc007: Rectangle, 60 pt wide, 24 pt high, border width 15 pt, black, dash size 0.5 pt
- 8 Rc008: Oval, based on default rectangle, bounding box 60 pt wide, 12 pt high, border width 1 pt, black
- 9 Rc009: Circle (oval), bounding box 40 pt wide, 40 pt high, border width 1 pt, black
- 10 Rc010: Circle (oval), bounding box 30 pt wide, 30 pt high, border width 5 pt, black
- 11 Rc011: Circle (oval), bounding box 12 pt wide, 12 pt high, border width 7 pt, black
- 12 Rc012: Oval, bounding box 60 pt wide, 84.75 pt high, border width 5 pt, black, dash size 5 pt

**Helpful Hint:** *The Rectangle element is a path. A filled rectangle can be simulated to some extent by overlaying the Rectangle element with a Cover element (see section 4.3.5.1) of the same size.*

### 4.3.5.3

### Covers

Cover elements are areas used to cover regions of a background image. They are always at the bottom of the element layers, and they cannot cover any other form elements.

The Cover tool  creates rectangular and oval cover areas. After selecting the tool a cursor with a cover in default size appears (see Fig. 4-30).

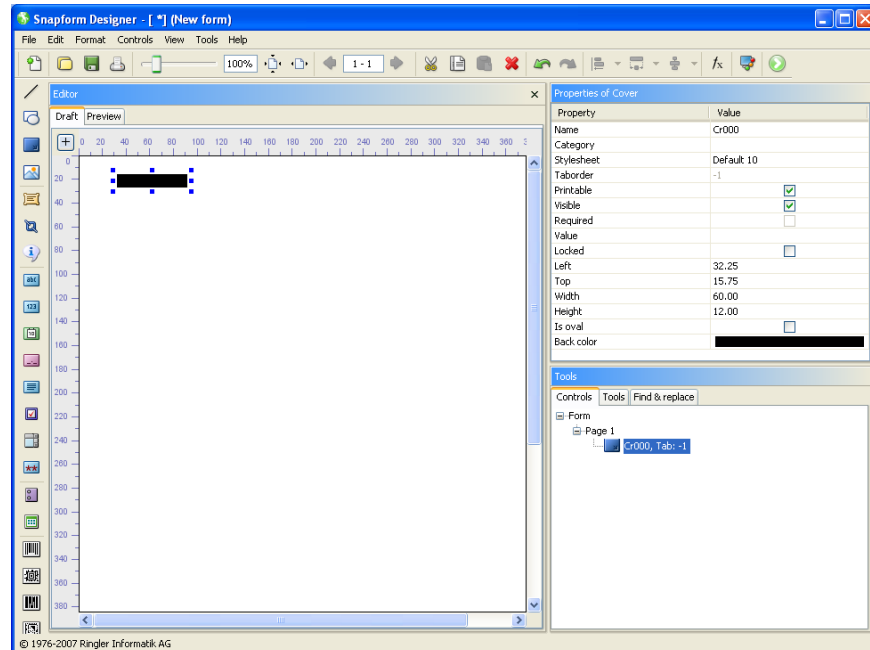
**Fig.4-30** Cover tool cursor



**Note:** *The cover area is represented in black so that it is easier to position it.*

Assigned to the Cover tool is a black rectangle, 60 pt wide and 12 pt high, with a line width of 1 pt. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Cr** and a three-digit consecutive number, beginning with **000**. The first cover placed in a form has therefore the name **Cr000**. In the object structure it is added to the according page. In Fig. 4-31 the first cover of the form has been placed and then selected.

**Fig. 4-31** *Newly created cover*



The length and the width of the cover area can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

**Note:** *When covers are superimposed, it is possible that the element placed later appears to be below the earlier placed element. The correct layering of the elements must be verified in the Preview mode.*

The other features of the cover are controlled in the Properties window (see Fig. 4-32).

**Fig. 4-32** *Properties of a cover*

Properties of Cover	
Property	Value
Name	Cr000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	32.25
Top	15.75
Width	60.00
Height	12.00
Is oval	<input type="checkbox"/>
Back color	

**Name**

The name of the cover in the document. This name can be changed.

**Category**

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

**Stylesheet**

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For covers, the stylesheet definition has however no relevance.

**Taborder**

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the

tabbing sequence, which is logical, because the Cover is not an active element.

**Printable**

When the box is checked, the cover will be printed.

**Visible**

When the box is checked, the cover is displayed on screen.

**Required**

This property is always deactivated, and cannot be changed.

**Value**

Clicking on this field opens the expression editor for assembling an expression. For Cover elements, an expression is however used rarely.

**Locked**

When this box is checked, the cover gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog become inactive. Access to these properties is resumed only after deactivating this check box.

**Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Cover.

**Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Cover.

**Width**

Width of the bounding box of the Cover element.

**Height**

Height of the bounding box of the Cover element.

**Is oval**

When this box is checked, an oval (ellipse), limited by the bounding box of the element is displayed instead of a rectangle.

## Back color

Color of the cover. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

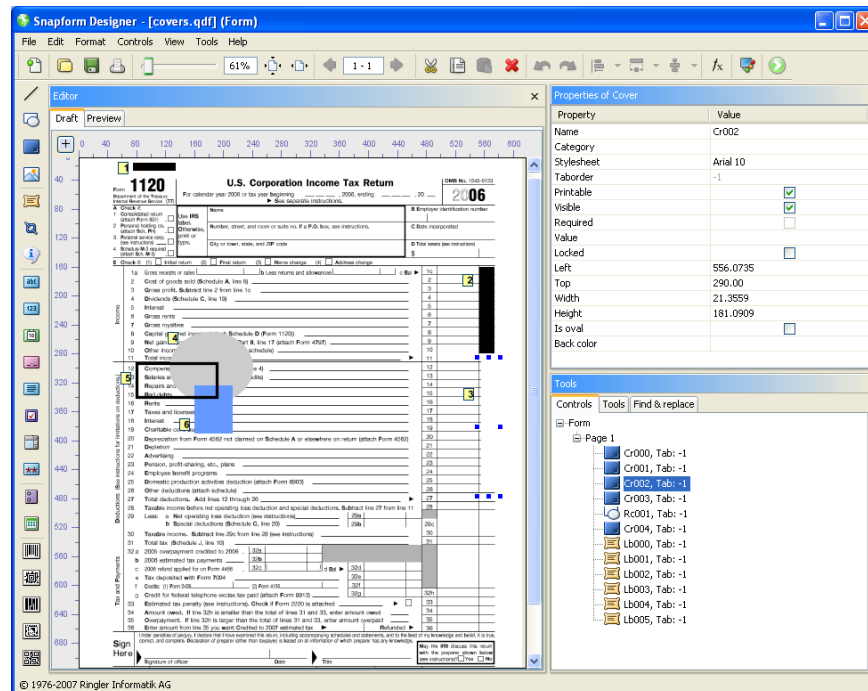
The document *covers.qdf* (see Fig. 4-33) shows various examples of Covers. After selecting a cover in the Snapform Designer, the properties window displays information about the features of that cover.

**Fig.4-33** Sample Cover elements in *covers.qdf*

In order to make all elements visible, Fig. 4-34 shows the same document in the draft mode of Snapform Designer. The actual Cover element (item 3 in the document) is selected. In addition, the window **Tools -> Controls** shows the element structure of the sample form.



**Fig. 4-34** Sample Cover elements in covers.qdf in Draft mode



- 1 Cr000: Default cover, 60 pt wide, 12 pt high, black
- 2 Cr001: Cover, rectangular, 21.35 pt wide, 120 pt high, black
- 3 Cr002: Cover, rectangular, 21.35 pt wide, 181.1 pt high, white
- 4 Cr003: Cover, oval, bounding box 115.9 pt wide, 99.6 pt high, light grey
- 5 Rc001: Rectangle, 114.9 pt wide, 51.2 pt high, border width 4 pt, black
- 6 Cr004: Cover, rectangular, 53.9 pt wide, 66 pt high, blue

Items 4, 5 and 6 (elements **Cr003**, **Rc001** and **Cr004**) demonstrate the overlapping behavior. Element **Cr003** is below element **Cr004** because the latter was added later. Element **Rc001** was added after element **Cr003**, but before element **Cr004**. Because Covers are always “at the bottom” in the element stack, the Rectangle element is on top of both Cover elements.

#### 4.3.5.4

### Images and logos

An important element of the base layout of forms are images and logos. Snapform allows inserting of raster graphics as images into an Image element, where they can be either bound statically into the document, or dynamically loaded (normally when the document is opened).

The advantage of a statically inserted image is that it is always available, independent on whether the Snapform Viewer can use a network connection, or whether the image is available locally.


Snapform can read most of the raster graphic formats used in an office or web environment. If a certain format is not supported, there are a large number of appropriate converters available. When everything fails, it is always possible to create a PDF and then extract the image using Acrobat (Professional).

It must be considered with raster images that there is a direct connection between the size of the image in pixels, the resolution of the image (in pixel per length unit; dpi, dots per inch) and the absolute dimensions of the image on the form (in inch or millimeters). These relationships cannot be changed.

A raster image always has a given size in pixels. This size is the result of the creation of the image. When the image is inserted into an Image element in Snapform, it gets a specific absolute dimension. This leads immediately to the resolution of the image in that particular case. This resolution is a measure of the quality in which the image can be displayed and printed. For color and greyscale images, the following resolutions are considered to be sufficient: 72 dpi for on screen, 150 dpi for office printers, 300 dpi for printing presses. For black/white images, the resolutions are 72 dpi for screen, 300 dpi for office printers, 1200 dpi for printing presses. These resolutions should not be exceeded too far, because the according display device cannot process considerably higher resolutions. The consequence of too high resolutions is an unnecessarily large file and long data transmission times.

Snapform Designer supports the following graphic formats for Image elements: JPEG, BMP, TIFF, GIF, and PNG.

**Note:** *JPEG is a very compact format, but uses a lossy compression which leads to artifacts and blurriness around high contrasts. High contrast occurs particularly with scanned text and line art. For this kind of images, the JPEG format is not suitable. Use TIFF, PNG or GIF instead.*

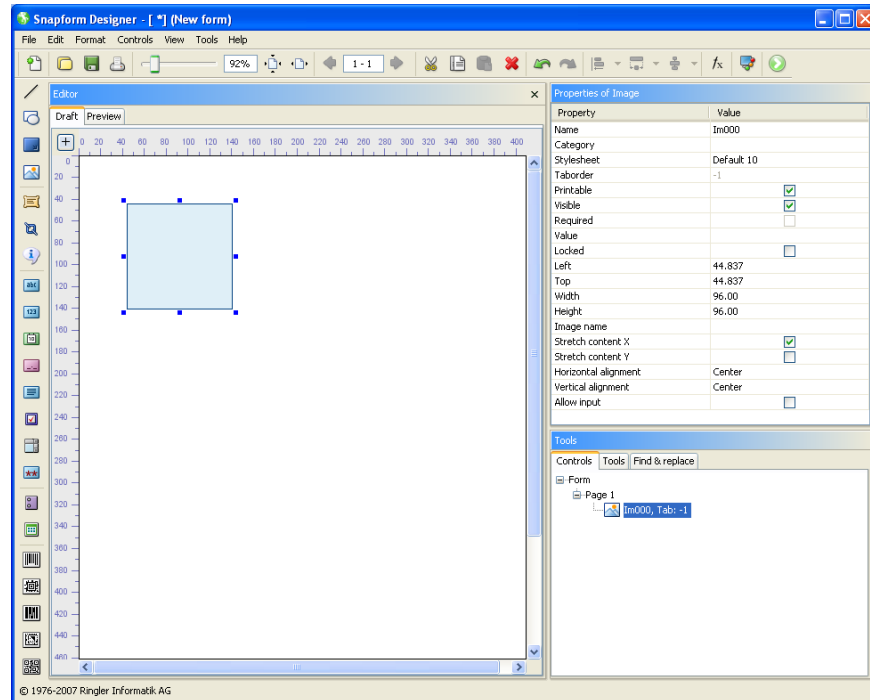
The Image tool  creates a field in which a raster graphic can be loaded. After selecting the tool, a cursor with a Image element in default size appears (see Fig. 4-35).

**Fig. 4-35** Image tool cursor



Assigned to the Image tool is an image field, 96 pt wide and 96 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Im** and a three-digit consecutive number, beginning with **000**. The first image field placed in a form has therefore the name **Im000**. In the object structure it is added to the according page. In Fig. 4-36 the first image field of the form has been placed and then selected.

**Fig. 4-36** *Newly created image field*



The length and the width of the image field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the image field are controlled in the Properties window (see Fig. 4-37).

**Fig. 4-37** *Properties of an image field*

Properties of Image	
Property	Value
Name	Im000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	44.837
Top	44.837
Width	96.00
Height	96.00
Image name	
Stretch content X	<input checked="" type="checkbox"/>
Stretch content Y	<input type="checkbox"/>
Horizontal alignment	Center
Vertical alignment	Center
Allow input	<input type="checkbox"/>

## Name

The name of the image field in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For image fields only the Border settings in the General settings tab are used.

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the Image is not an active element.

## Printable

When the box is checked, the image field will be printed.

## Visible

When the box is checked, the image field is displayed on screen.

## Required

This property is always deactivated, and cannot be changed.

## Value

Clicking in the property field opens the expression editor for entering an expression.

In image fields with referenced contents (meaning that the image is not directly inserted into the field), the result of the expression is the path to the image file to be inserted. The path is an URL (Uniform Resource Locator), which in the case of images is mainly one of the following protocols: *http*, *https*, *ftp*, *file*. In order for the form to use the first three protocols, an Internet connection must be available and active. The file: protocol refers to a local file which means that the path to this file must either be the result of a calculation, or constant.

The simplest example is the definition of a constant value (item 2 in the sample document *images.qdf*, Fig. 4-39):

```
http://www.google.ch/images/logo_sm.gif
```

This is a constant value, and when the form opens, the specified image is loaded from the Internet.

Rather similar is (not shown in the sample document) a constant result:

```
= "http://www.google.ch/images/logo_sm.gif"
```

This loads the specified image when the form opens, and each time when a general recalculation occurs.

It is also possible to set up an indirect reference. In this case, the URL is stored in a text field, and its value is used in the calculation (item **3** in the sample document *images.qdf*, see Fig. 4-39). The expression for this is:

```
=Text000
```

Where the URL must be the value of the text field **Tx000**.

The result of a slightly more complex expression is the reference via a check box (item **5** in the sample document *images.qdf*, see Fig. 4-39). The expression for this is:

```
=if(Cb000, "http://www.snapform.com/img/s-blue.jpg",  
"http://www.snapform.com/img/s-green.jpg")
```

When the check box **Cb000** is checked, the file *s-blue.jpg* is loaded (the blue Gecko), otherwise, the file *s-green.jpg* is loaded (the green Gecko).

Even more sophisticated logic is demonstrated in item **6** of the sample document *images.qdf*.

## Locked

When this box is checked, the image field is locked in the Snapform Designer, and cannot be (accidentally) moved. In addition, all other properties in the properties dialog are made inactive. Access to these properties is only available after unchecking this box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the image field.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the image field.

## Width

Width of the image field in Points.

## Height

Height of the image field in Points.

## Image name

When the image is to be directly inserted into the image field, its name is specified in this property field. Clicking in this field opens a drop-down menu with the names of the images already inserted into the document. Clicking on the ... button opens a normal File Open dialog in which the image file can be specified.

An image specified via this property is stored within the document and can be reused in other image fields. A typical use of this feature is for logos appearing on every page of the document.

## Stretch content X and Stretch content Y

Only in rare cases, the inserted image has exactly the same size as the image field. These check boxes control whether the image is displayed in its original size, or whether it will be scaled to fit into the image field in the according direction. The following combinations are possible:

- Both boxes unchecked:

The image is displayed in its original size. Parts of the image not fitting into the image field are cut off.

- **Stretch content X** checked and **Stretch content Y** unchecked:

The image is scaled proportionally so that it completely fits into the image field. When it is proportionally wider than the image field, the width is the same as of the image field.

- **Stretch content X** unchecked and **Stretch content Y** checked:

The image is scaled proportionally so that it completely fits into the image field. When it is proportionally taller than the image field, the height is the same as of the image field.

- Both boxes checked:

The image is scaled in both directions that it fills out the image field. The proportions of the image are changed.



## Horizontal alignment

When the image is narrower than the image field, this property defines how it is horizontally positioned within the field.

When the property field is clicked, a drop-down menu appears with the options **left**, **center**, **right**, which places the image left-aligned, centered or right-aligned within the image field. Default is **center**.

## Vertical alignment

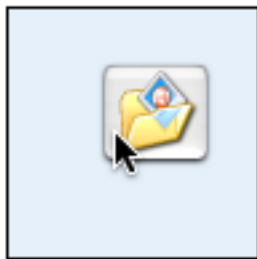
When the image is shorter than the image field, this property defines how it is vertically positioned within the field.

When the property field is clicked, a drop-down menu appears with the options **top**, **center**, **bottom**, which places the image top-aligned, centered or bottom-aligned within the image field. Default is **center**.

## Allow input

When this box is checked, the user can insert his own image into the field when filling out the form. A typical application for this is inserting a passport picture in a personnel form. Item **4** (field Im005) in the sample document *images.qdf* is such an image field. When the cursor hovers over such a field, it changes to the image inserting cursor (see Fig. 4-38).

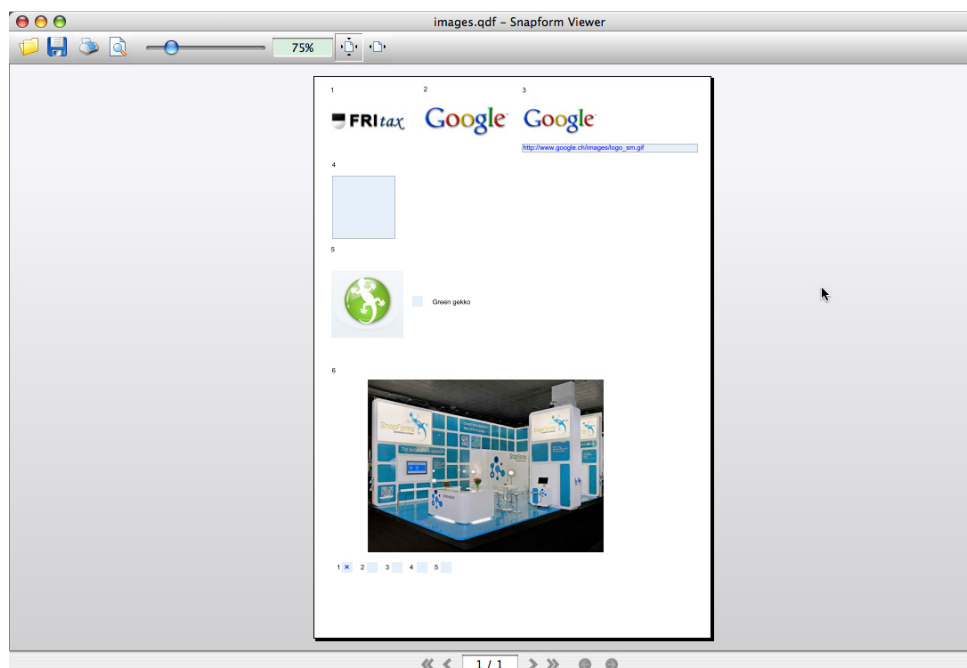
**Fig.4-38** Image inserting cursor



When the user clicks on this field, a File Open dialog opens to select the image file to be inserted. This image will then be included in the document, and remains there after saving.

The document *images.qdf* (see Fig. 4-39) shows various examples of Images. After selecting an image field, the properties window and the expression editor display information about the features of that image field.

**Fig. 4-39** *Sample image fields from images.qdf*



- 1 Im000: Image field, 113.5 pt wide, 60.7 pt high, image start\_panel\_top.png inserted into document
- 2 Im001: Image field, 129.3 pt wide, 63.7 pt high, URL of image defined as a constant
- 3 Im003: Image field, 113.6 pt wide, 60.7 pt high, URL of image defined in text field Tx000; indirect reference
- 4 Im005: Image field, 96 pt wide, 96 pt high, user input allowed
- 5 Im002: Image field, 109.7 pt wide, 133.2 pt high, URL controlled via check box Cb000
- 6 Im004: Image field, 465.3 pt wide, 260 pt high, URL controlled via set of 5 radio buttons

The expressions for items **2**, **3** and **5** are listed above. They can also be displayed in the expression editor by double-clicking on the respective field in the Draft mode.

## 4.3.6

### Text

Text is an important part of forms. Text belongs partly to the base layout and partly to the Form layer. This section covers text belonging to the base layout.

In order to display text in the base layout, the Label element is used. Label elements may contain simple or formatted text.

**Note:** *In earlier Snapform versions, Labels were, besides displaying text, also used as a container for expressions. This kind of use is considered to be obsolete, and should no longer be implemented.*

Simple (unformatted) text consists of one single font and text size, with one single attribute throughout the field. This kind of text can be entered as simple text, or as HTML. The HTML form is necessary for certain attributes, such as vertical alignment in the field, or text attributes which cannot be set in the properties window of the element.

Formatted text may consist of several font sizes and types, and have changing attributes. Formatted text is entered in HTML form. All HTML tags not using external resources are supported. External stylesheets are not supported either (see also section 4.3.6.2).


**Note:** *When using text in HTML form, be aware that the rendering mechanism for HTML text has a lower resolution than for simple text. It is possible that text entered in HTML form may not be of the expected quality level when printed out.*

**Helpful Hint:** *Entering text in HTML form into the expression editor can sometimes be rather tedious. It is therefore recommended to use an external text editor which supports HTML formatting in a more suitable way.*

*An additional help for text entry is provided by utilities such as Markdown, which use a simplified syntax for creating formatted text and then convert it in the text editor to HTML code.*

### 4.3.6.1

## The Label tool

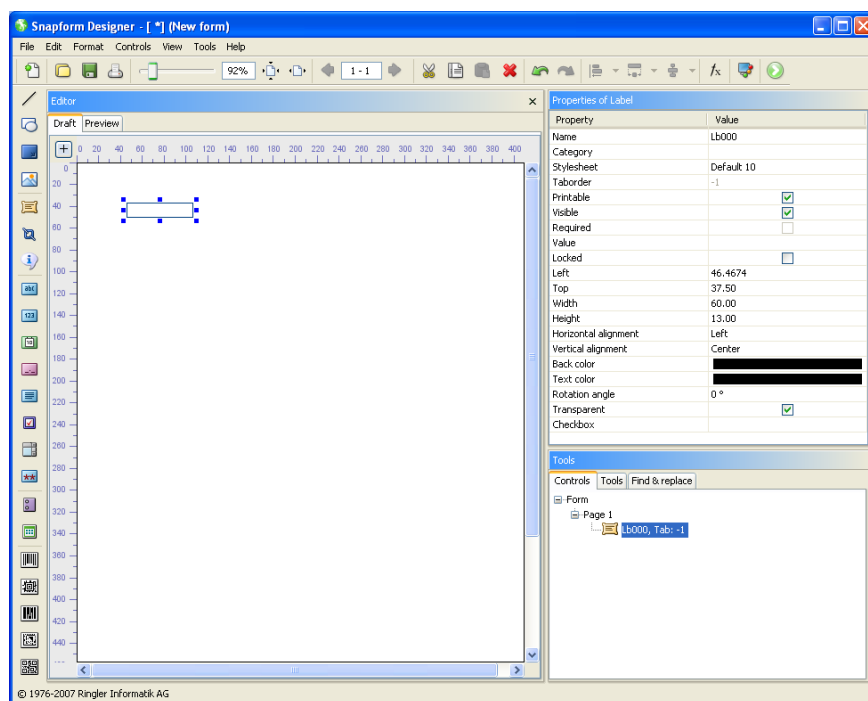
The Label tool  creates a field for displaying text. After selecting the tool, a cursor with a text frame in default size appears (see Fig. 4-40).

**Fig. 4-40** *Label tool cursor*



Assigned to the Label tool is a rectangle, 60 pt wide and 13 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Lb** and a three-digit consecutive number, beginning with **000**. The first label placed in a form has therefore the name **Lb000**. In the object structure it is added to the according page. In Fig. 4-41 the first label of the form has been placed and then selected.

**Fig. 4-41** *Newly created label*



The length and the width of the label can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor

point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the label are controlled in the Properties window (see Fig. 4-42).

**Fig.4-42** *Properties of a label*

Properties of Label	
Property	Value
Name	Lb000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	46.4674
Top	37.50
Width	60.00
Height	13.00
Horizontal alignment	Left
Vertical alignment	Center
Back color	
Text color	
Rotation angle	0 °
Transparent	<input checked="" type="checkbox"/>
Checkbox	

**Name**

The name of the label in the document. This name can be changed.

**Category**

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For labels the zones **Font settings** and **Supplemental font support** in the **General settings** tab of the stylesheet editor are of importance.

**Note:** *These stylesheet definitions are in no context with Cascading Style Sheets in HTML, and do not have any effect on the representation of text in HTML form.*

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the Label is not an active element.

## Printable

When the box is checked, the label will be printed.

## Visible

When the box is checked, the label is displayed on screen.

## Required

This property is always deactivated, and cannot be changed.

## Value

The text to be displayed is entered in this property of the Label element. Clicking on this property field opens the expression editor.

Unformatted text is entered directly, without quotes or equal sign. HTML code can also be entered directly. If the text to be displayed is the result of a calculation (for example the combination of constant text with a field value), an equal sign is necessary, and the text (including HTML code, if present) must be set between double quotes.

HTML code is, if it is not the result of calculations or not too complex, directly interpreted and rendered even in Draft mode. When the code is more complex (tables, images) or the result of a calculation, the text

cannot be displayed directly. This kind of text requires viewing the form in the Preview mode to display correctly. In this case, the label field shows the expression when in Draft mode.

Further examples of labels with text in HTML form can be found in section 4.3.6.2.

## Locked

When this box is checked, the label is locked in the Snapform Designer, and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after unchecking this check box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the label.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the label.

## Width

Width of the bounding box of the Label element.

## Height

Height of the bounding box of the Label element.

## Horizontal alignment

This property defines how the text is horizontally aligned in the label.

When the property field is clicked, a drop-down menu appears with the options **left**, **center**, **right**, which places the text left-aligned, centered or right-aligned within the label. Default is **left**.

With unformatted text with multiple lines, each line is aligned as specified. With the HTML form, the longest line of a text block is aligned as specified. All other lines are then left-aligned to this reference line.

## Vertical alignment

This property defines how the text is vertically aligned in the label.

When the property field is clicked, a drop-down menu appears with the options **top**, **center**, **bottom**, which places the text top-aligned, centered or bottom-aligned within the label. Default is **center**.

This property is fully honored only with text in HTML form. In this case, the HTML text block gets aligned.

Single line unformatted text is always aligned to the bottom border of the field. Multi-line unformatted text is always aligned to the top border of the field.

### Back color

Background color of the label. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

The background color is only applied when the **Transparent** check box is unchecked.

### Text color

Color of the text. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

### Rotation angle

The text in a label can be rotated by multiples of 90°. Clicking on this property opens a drop-down menu with the selection of the rotation angles 0°, 90°, 180°, 270°. The default value is 0°.

### Transparent

The background of the label is set to transparent (no covering background color) with this check box. This property is active by default.

### Check box

It is possible to associate the label with a check box. The label becomes clickable and an “extension” of the check box. When the label is clicked, the state of the check box changes.

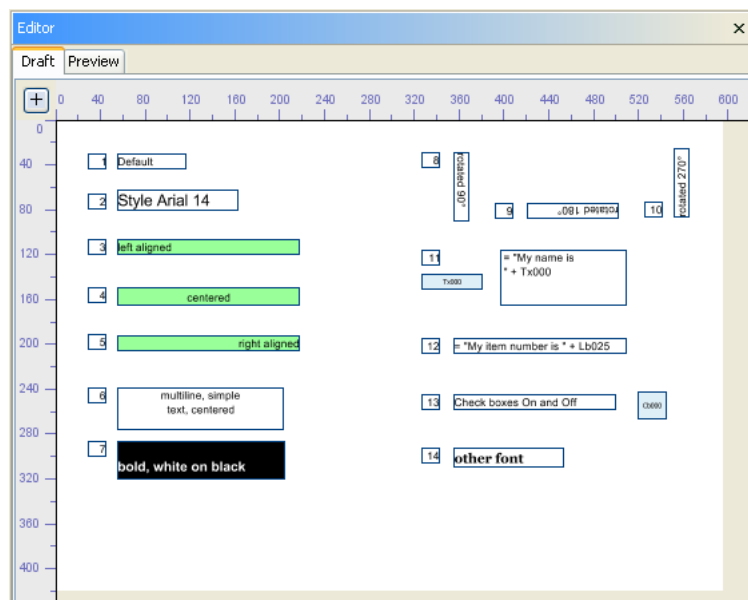
Clicking on this property opens a drop-down menu with the selection of all check boxes in the document. Selecting the according field name sets the association.



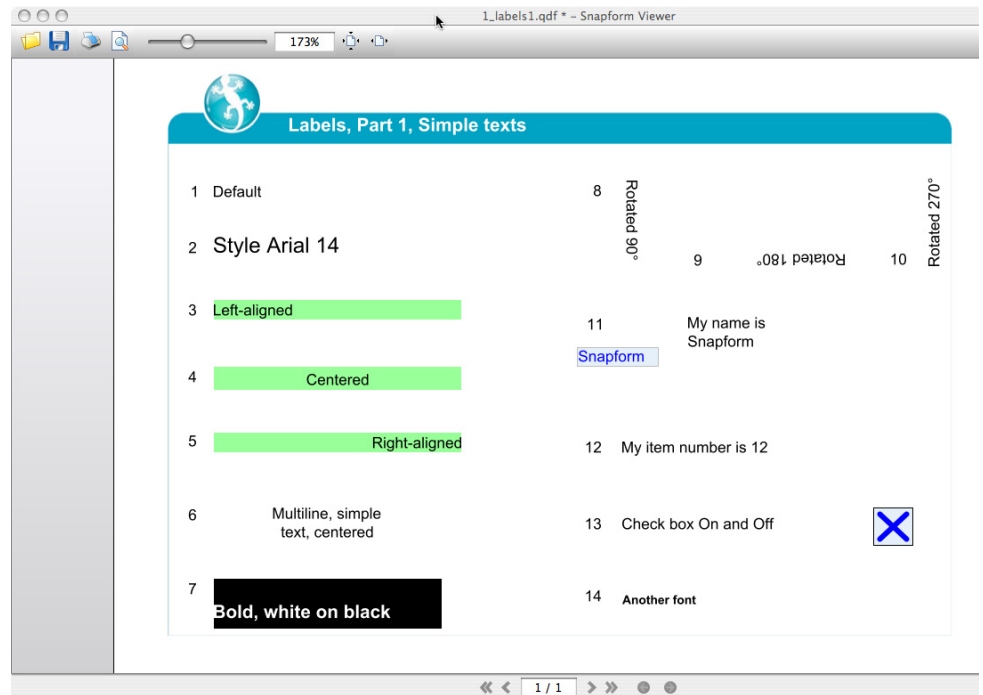
The document *labels1.qdf* (see Fig. 4-43 (Draft mode) and Fig. 4-44 (shown in the Snapform Viewer)) shows various examples of labels with simple text. The document *labels2.qdf* (see Fig. 4-45 (Draft mode) and Fig. 4-46 (displayed in the Snapform Viewer)) shows examples of labels with formatted (HTML) text.

After selecting the label in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the text.

**Fig. 4-43** Label examples of *labels1.qdf* in Draft mode

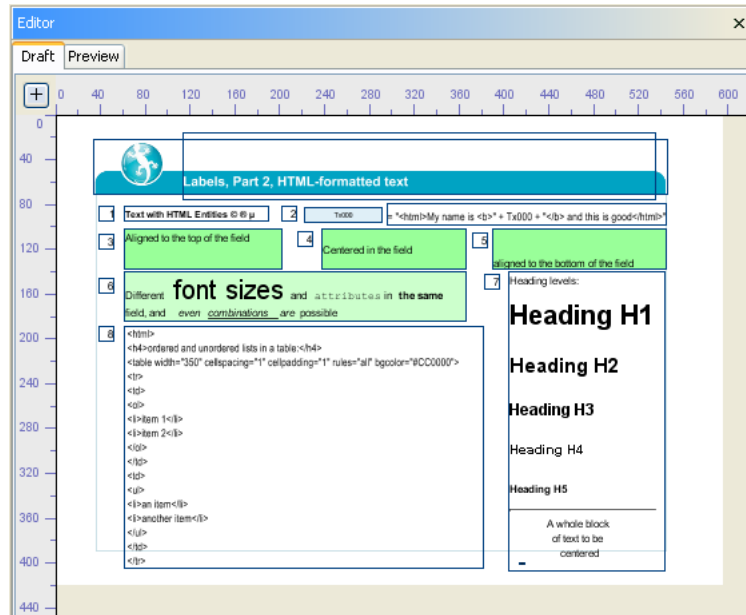


**Fig. 4-44** Label examples of labels1.qdf in the Snapform Viewer

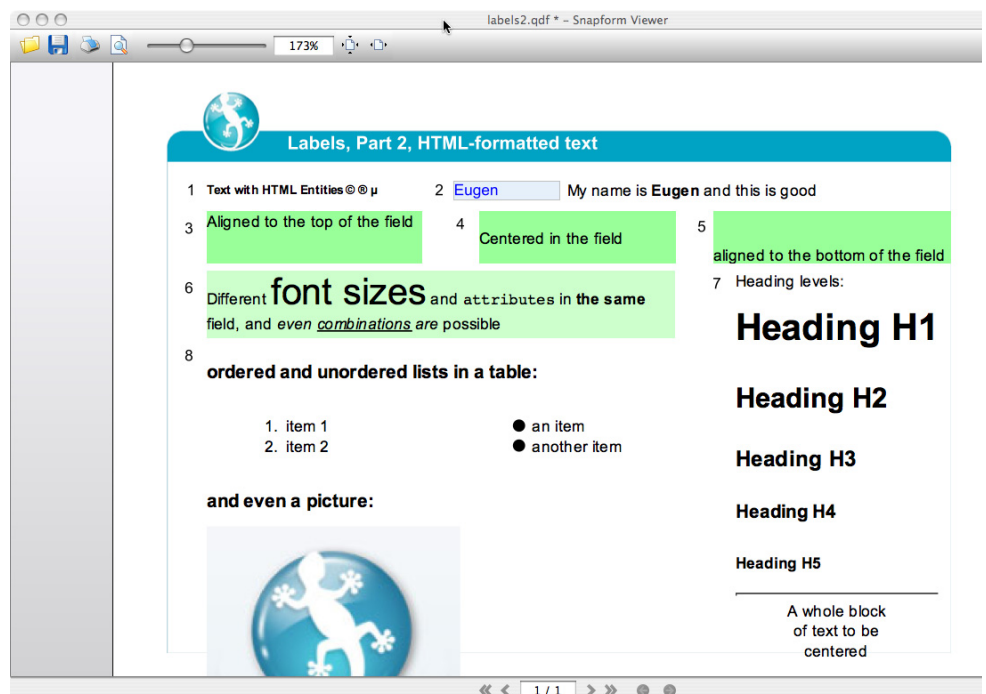


- 1 Lb000: Label, default settings
- 2 Lb001: Label, font size specified in stylesheet
- 3 Lb002: Label, text left-aligned, background color green (to visualize the field size), non-transparent
- 4 Lb003: Label, text centered, background color green (to visualize the field size), non-transparent
- 5 Lb004: Label, text right-aligned, background color green (to visualize the field size), non-transparent
- 6 Lb005: Label, multiline text, aligned to the top border of the field
- 7 Lb006: Label, font bold specified in stylesheet, single-line text aligned to the bottom border of the field
- 8 Lb007: Label, text rotated by 90°
- 9 Lb008: Label, text rotated by 180°
- 10 Lb009: Label, text rotated by 270°
- 11 Lb010: Label, text consists of constant part and the value of field Tx000; the value of field Tx000 is transferred when leaving the field
- 12 Lb011: Label, text consists of constant part and the value of the label Lb025
- 13 Lb012: Label, associated with check box Cb000; the label becomes active and clicking changes the state of the check box
- 14 Lb013: Label, font type specified in stylesheet

**Fig. 4-45** Label examples of labels2.qdf in Draft mode



**Fig. 4-46** Label examples of labels2.qdf in the Snapform Viewer



- 1 Lb000: Label, text in HTML, with special characters as HTML entities
- 2 Lb007: Label, text consists of constant part and the value of field Tx000; the value of field Tx000 is transferred when leaving the field,

and it is displayed in boldface

- 3 Lb001: Label, text in HTML, aligned to the top border of the field, background color green (to visualize the field size), non-transparent
- 4 Lb002: Label, text in HTML, aligned to the center of the field, background color green (to visualize the field size), non-transparent
- 5 Lb003: Label, text in HTML, aligned to the bottom border of the field, background color green (to visualize the field size), non-transparent
- 6 Lb004: Label, text in HTML, with various changing in-line text attributes
- 7 Lb005: Label, text in HTML, with Heading levels, horizontal separator line and centered text
- 8 Lb006: Label, text in HTML, with lists, tables and embedded image

#### 4.3.6.2

#### HTML tags and attribute usable in Snapform

Snapform supports tags and attributes which are defined in the HTML 3.2 standard. Newer features are not supported. Such code may either be displayed as code, or not at all. The official HTML specification can be found at **<http://www.w3c.org/TR/REC-html32>**.

## 4.4 Form layer

One level above the base layout is the form layer. The form layer consists of active elements, which means that these elements react to user interaction (such as a value can be entered into a field, or that a box can be checked).

In very simple words, part of the form layer are elements whose tab order value is greater than 0, and can be changed. This includes the Image element (see section 4.3.5.4). Because the interactivity of this element is rarely used, it has been assigned to the base layout.

On the other hand, labels which are normally part of the base layout, function as an expression container and are also considered to be part of the form layer.

The elements of the form layer can be saved on their own. The file format uses the extension **.hdo**. In order to save the form layer, select the option **Snapform field layer (.hdo)** in the Save as dialog. A saved field layer is loaded with the menu item **File -> Open Field layer**.

### 4.4.1 Labels and expression container

Labels and expression containers are the same, but named differently according to their use. In earlier Snapform versions, only labels were implemented to hold expressions. In the current version, expressions can be assigned to any element of the form, and functions are valid for the whole document. Because of this, the purpose of labels as expression containers has only historical purposes.

**Note:** *It is recommended to overhaul forms using labels as expression containers, and migrate to functions instead (more about functions in section 4.5.1.2).*

## 4.4.2

## Entry fields

One of the fundamental purposes of a form is the data acquisition. Data is normally acquired via entry fields. Snapform uses text fields in their widest sense, option fields and check boxes, as well as lists to acquire data. Text fields are separated into specialized field types (such as Date or Entry masks).

The following sections describe first the common features of text fields, and then the specifics of the different “special purpose” fields.

### 4.4.2.1

### Entry fields in general

Entry fields are placed on the workspace after selecting the according tool. They are added to the document structure and inserted into the taborder. The properties shown in Fig. 4-47 and explained below are the same for all entry fields (except some properties which are not available for Password fields).

**Fig. 4-47** *Properties of entry fields shown with a Text field*

Properties of Text field	
Property	Value
Name	Tx000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	68.4783
Top	40.7609
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Horizontal alignment	Left
Length	0
Configure boxing	...
Link to object	
Only digits	<input type="checkbox"/>

## Name

The name of the entry field in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For entry fields the **General**

**settings** and **Input controls and labels** tabs in the stylesheet editor have been made for the presentation of different states of entry fields.

## Taborder

Sequence number of the element in the tab order. This value is assigned automatically, and is in the order of the field tab sequence (beginning with 10 and incremented in steps of 10). The first field of the form has tab order value 10, the second the value 20, etc. The tab order value can be changed, so that a specific tabbing sequence can be created which has no relationship with the order the fields have been added.

## Printable

When the box is checked, the entry field will be printed.

## Visible

When the box is checked, the entry field is displayed on screen.

## Required

When this box is checked, the entry field must have a value entered in order for the form to be valid (more about validity of the form, see section 4.8.4.2).

## Value

This property can be used twofold for entry fields. First, the default value of the field can be specified (by entering a simple value); second, an expression can be specified whose result is displayed in the field.

A simple value is the default value which is displayed when the blank form is opened, and it can be overwritten when filling out the form. A simple value looks in the expression editor like this:

**This is the default value**

Resetting the form (Clear entries button in the Snapform Viewer) redisplayes the default value.

A default value specified as an expression:

**= "Default value as expression"**

is displayed as such when opening the document. When it is overwritten as the form is filled out, the field will be marked as



“calculated field - overwritten”. When the form is reset, the field is cleared, but it will remain marked as “overwritten”. Only after clicking the field’s reset icon, the default value is shown again.

**Note:** *Entry fields do not support text in HTML form.*

An expression is executed when the document is opened, and when form actions are run. When the field is not write-protected, the field value can be overwritten. The field will however be marked as “overwritten”.

### **Locked**

When this box is checked, the entry field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

### **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the entry field.

### **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the entry field.

### **Width**

Width of the bounding box of the entry field.

### **Height**

Height of the bounding box of the entry field.

### **Read-only**

When this box is checked, the entry field is marked as read-only, and its content cannot be manually changed. The value may still be changed as the result of an expression. A read-only entry field is still part of the tab order, and its content may be copied to the clipboard with **<Ctrl><C>** respectively **<Cmd><C>**.

**Note:** *This property is not available for Password fields.*

## Single use field

There is information which may no longer be modified after initially filling out the form, and whose fields must therefore be protected. In Snapform, this protection occurs in Single use fields, which can be modified until the form is saved for the first time. After that, the field is marked as read-only.

When this box is checked, the entry field can be modified until the form is saved for the first time. After saving for the first time, the entry field will be marked as read-only.

**Note:** *This property is not available for Password fields.*

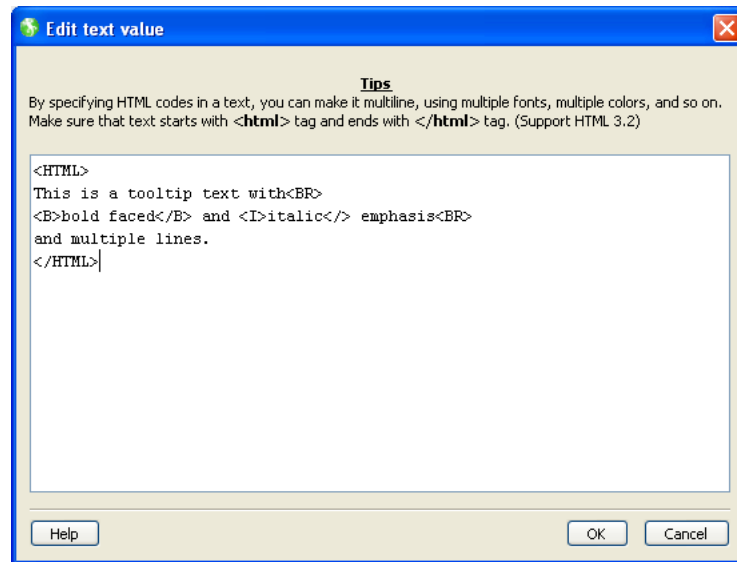
## Tooltip text

The Tooltip text is help text that appears when the mouse cursor hovers over the field for a certain time. This text is in many cases the first and simplest help to the according field. With increased requirements for accessibility, the tooltip text becomes more and more important because screen readers use this information. It is particularly important for government forms, that the form designer inserts well thought out tooltips.

Double-clicking on this property opens the **Edit text value** window (see Fig. 4-48). As indicated in the explanatory text, it is also possible to enter text in HTML form, for which the same rules apply as for labels.

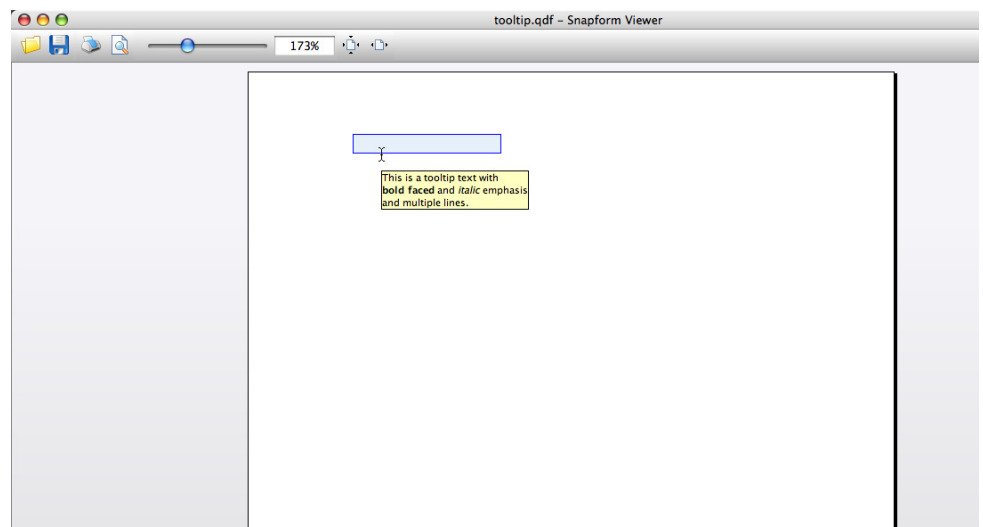
**Note:** *Tooltip text should be short. Therefore it is advised to avoid entering excessive HTML formatting and inserting images.*

**Fig. 4-48** *Edit text value window*



The tooltip text defined in Fig. 4-48 as HTML text appears in the Snapform Viewer as shown in Fig. 4-49.

**Fig. 4-49** *Displaying the tooltip text*



## Horizontal alignment

This property defines how the text is horizontally aligned in the entry field.

When the property field is clicked, a drop-down menu appears with the options **left**, **center**, **right**, which places the text left-aligned, centered or right-aligned within the entry field. Default is **left**.

Multiline text fields and Password fields do not honor this property.

## Configure boxing

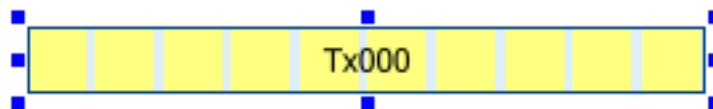
This property boxes the entry field (and creates, for example, a comb field). A boxed field is a field where single characters or groups of characters are placed together into boxes which have a certain distance from each other. This kind of fields is often used for evaluations in paper-based workflows where forms are filled out by hand, and it is said to be friendly to OCR systems.

As opposed to the frequently used method to create fields of their own for individual boxes, segmented fields allow entering a value “in one piece”, which is a necessity for electronic forms (such as for further calculations or connections to a back-end system).

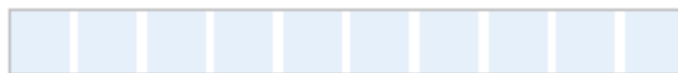
Clicking on ... in this property opens the configuration window. This window is specific for each different field type and explained in the according section.

When an entry field gets boxed using this property, its appearance changes in Draft mode (see Fig. 4-50), as well as in the Preview mode (see Fig. 4-51).

**Fig. 4-50** *Boxed entry field in Draft mode*



**Fig. 4-51** *Segmented entry field in Preview mode*



Multiline text fields and Password fields do not honor this property.

## Link to object

In many forms applications, there are references to other sections of the form, or to additional information. With the **Link to object** property it is possible to create such a reference in a Snapform form.

When this property is selected, a drop-down list containing all the elements of the form layer (all elements with a taborder value greater than 0, including Image fields), which can be selected. When the form is

filled out, a green arrow mark appears at the right of the field (see Fig. 4-52). Clicking on this icon sets the focus to that referenced field.

**Fig.4-52** *Arrow icon besides a field with Link to object*




The other properties specified in the Properties window are specific for the actual type of entry field and are explained in the appropriate sections.

#### 4.4.2.2

#### Text fields

Text fields are entry fields for any kind of unformatted text which fits on a single line (for multiline text fields see section 4.4.2.3).

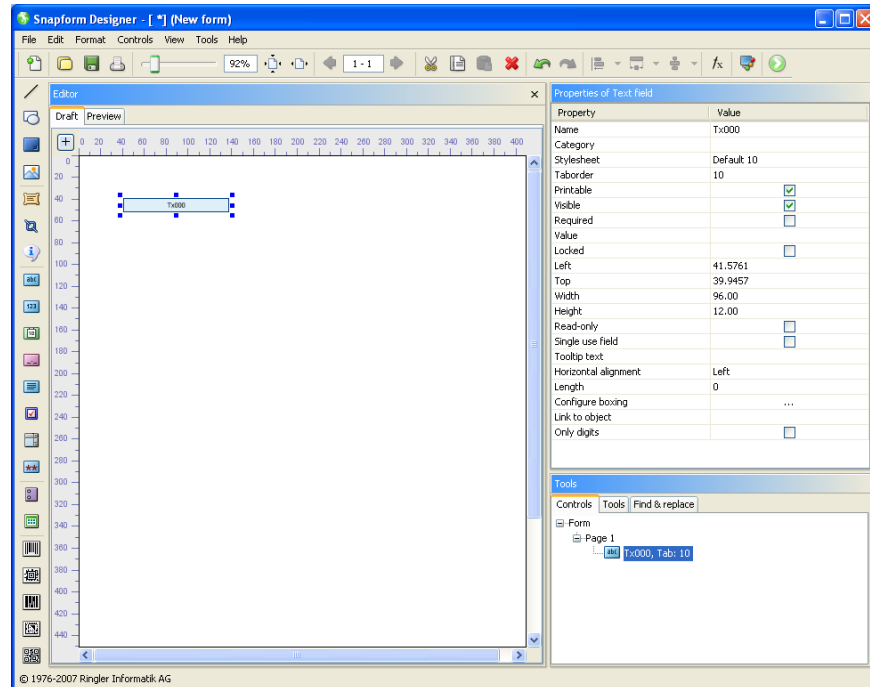
The Text field tool  creates a field which allows displaying and entering text. After selecting the tool, a cursor with a Text field element in default size appears (see Fig. 4-53).

**Fig.4-53** *Text field cursor*



Assigned to the Text field tool is a text field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Tx** and a three-digit consecutive number, beginning with **000**. The first text field placed in a form has therefore the name **Tx000**. In the object structure it is added to the according page. In Fig. 4-54 the first text field of the form has been placed and then selected.

**Fig. 4-54** *Newly created text field*



The length and the width of the text field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the text field are controlled in the Properties window (see Fig. 4-55).

**Fig. 4-55** *Properties of a text field*

Properties of Text field	
Property	Value
Name	Tx000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	41.5761
Top	39.9457
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Horizontal alignment	Left
Length	0
Configure boxing	...
Link to object	
Only digits	<input type="checkbox"/>

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for text fields, or have additional information pertinent to the description of the common properties.

## Value

It is not possible to enter HTML-formatted text.

## Length

Maximum number of characters in the field. When this number of character is reached, any further entered characters are ignored. No value or the value 0 mean that the field has no length limitation.

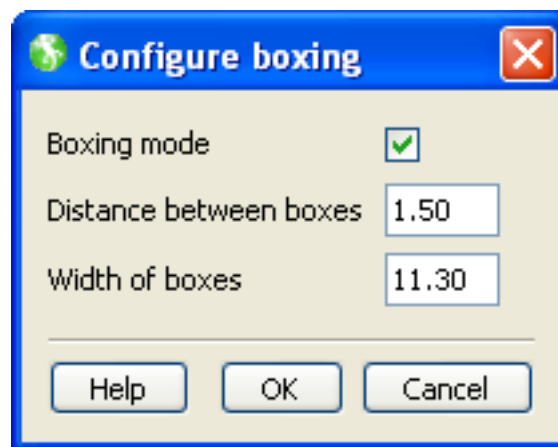
When filling in the form, text that is too long is displayed in a smaller font size and with condensed characters, so that it completely fits into the field. The consequence is that text can become unreadable rather quickly. It is therefore important to estimate the text size when planning the form, and to provide text fields with an appropriately sized data entry field.

## Configure boxing

Text fields are boxes into individual characters. This kind of boxed fields is also called “comb fields”.

Clicking on ... for this property opens the field boxing configuration window (see Fig. 4-56).

**Fig.4-56** Configuration window for field boxing



The configuration window for field boxing allows to set the following properties:

## Boxing mode

This check box activates and deactivates the field boxing. When the box is checked, the field gets boxed, when it is unchecked, the field does not get boxed.



## Distance between boxes

The distance between the boxes for the individual characters, measured in Points. Default value is 1.5 pt.

## Width of boxes

The width of a box for the individual characters measured in Points. Default value is 11.3 pt.

**Note:** *As the default value for the width of boxes is not connected to the font size defined in the stylesheet, it is important to make sure that the width of the boxes is sufficient when larger fonts sizes are used. As a rule of thumb use the value of the font size of the field for just numbers, and 1.2 times the value of the font size of the field for general text.*

When the field boxing is activated, the width of the entry field is recalculated, based on the number of characters specified in the **Length** property, and the field is scaled accordingly. If no length has been specified, or its value is 0, the field gets automatically scaled to a width for 10 characters. This value will also be entered into the **Length** property.

## Only digits


When this box is checked, only numbers between 0 and 9 can be entered. All other entries are ignored.

The difference between this and a numeric field is that preceding zeros are not suppressed. This property is often used for serial numbers or similar information.

### 4.4.2.3

## Multiline fields

To display or enter text when the single line text field tool is not sufficient. In this case, the multiline text tool can be used, which allows a unlimited number of lines.

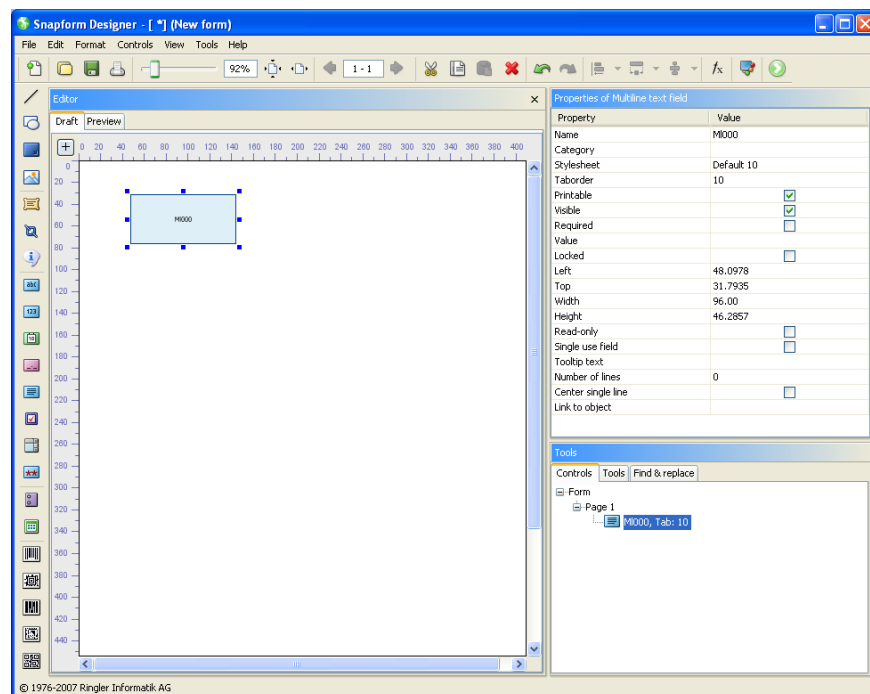
The Multiline text field tool  creates a field which allows displaying and entering multiple lines of text. After selecting the tool, a cursor with a Multiline text field element in default size appears (see Fig. 4-57).

**Fig. 4-57** Multiline text field  
cursor



Assigned to the Multiline text field tool is a multiline text field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **MI** and a three-digit consecutive number, beginning with **000**. The first multiline text field placed in a form has therefore the name **M1000**. In the object structure it is added to the according page. In Fig. 4-58 the first multiline text field of the form has been placed and then selected.

**Fig. 4-58** Newly created  
multiline text field



The length and the width of the multiline text field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the multiline text field are controlled in the Properties window (see Fig. 4-55).

**Fig.4-59** *Properties of a multiline text field*

Properties of Multiline text field	
Property	Value
Name	Ml000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	48.0978
Top	31.7935
Width	96.00
Height	46.2857
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Number of lines	0
Center single line	<input type="checkbox"/>
Link to object	

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for multiline text fields, or have additional information pertinent to the description of the common properties.

## Value

It is not possible to enter HTML-formatted text.

## Number of lines

The maximum number of lines which the field can handle. When the value is set to 0, there is no limitation in the number of lines.

**Attention:** *When the number of lines is not limited, and the bottom of the field has been reached, any further entry occurs “blind”. This text cannot be displayed, and will not be printed either. Since there are no scroll bars in the field, that text remains invisible. When data is submitted, the entire text gets transferred. In order to make sure that all the text is displayed to the user, the number of lines must therefore be limited.*

**Note:** *A multiline text field does not insert automatic line breaks. The text will therefore be scaled in the same way as with regular text fields. Therefore, make sure that when entering text, manual line breaks are inserted.*

## Center single line

This property controls the behavior of the field when only one single line of text is entered. Normally, and also when entering text, it is aligned to the top border of the field. When this box is checked, the text is centered, even if it consists of only one line. This ensures that with fields with a low number of lines, particularly in tables, a more harmonic presentation of the text.

### 4.4.2.4

## Numeric fields

Numbers can also be entered into regular text fields. Numeric fields offer, however, a wider range of controlling the entries. In addition, numeric fields display numbers normally right-aligned which simplifies the work with tabular forms.


Numeric fields accept only the numbers from 0 to 9, the decimal character (according to the language version of the Snapform Designer and the Locale settings of the computer), and the minus sign.

The decimal sign is stored internally as a symbolic “decimal sign”. This ensures that it gets displayed correctly according to the language settings. A form saved in a German environment will be correctly displayed when opened in an English environment, and it behaves properly.

The minus sign behaves like a toggle switch. Whenever it is pressed when entering a number, it toggles the number from positive to

negative and vice-versa. The minus sign does not need to be entered at the beginning of the entry. The minus sign, too, is stored internally as symbolic “minus sign”, and it is displayed according to the environment settings.

**Note:** *Numeric fields have a limited number of digits which can be processed. This limit is at 16 significant digits, and has technical reasons (practically all available software has this limitation too). If within calculations more than 16 significant digits are used, small errors may propagate to subsequent results.*

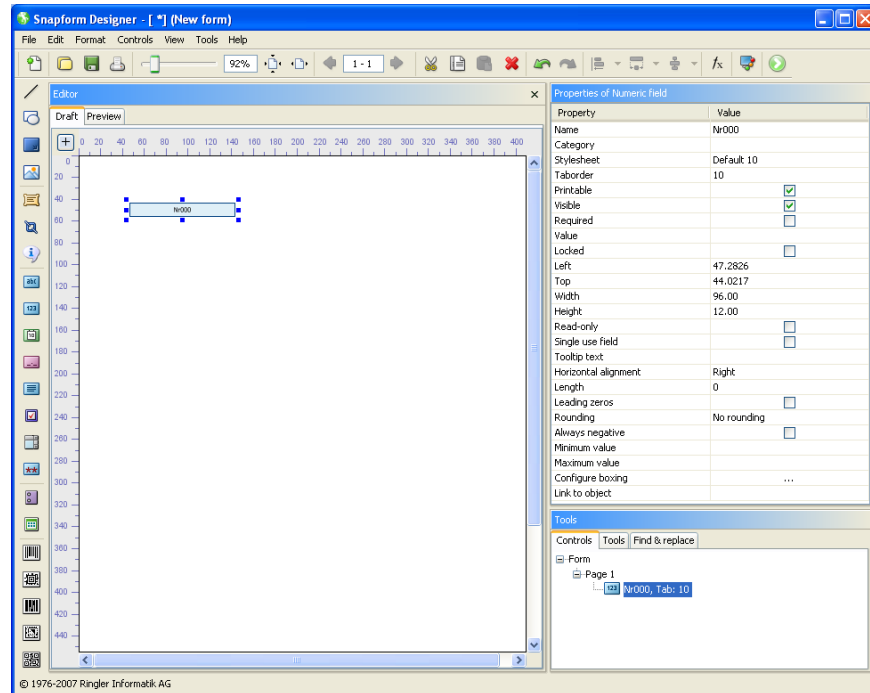
The Numeric field tool  creates a field which allows displaying and entering numbers. After selecting the tool, a cursor with a Numeric field element in default size appears (see Fig. 4-60).

**Fig. 4-60** *Numeric field cursor*



Assigned to the Numeric field tool is a numeric field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Nr** and a three-digit consecutive number, beginning with **000**. The first numeric field placed in a form has therefore the name **Nr000**. In the object structure it is added to the according page. In Fig. 4-61 the first text field of the form has been placed and then selected.

**Fig. 4-61** *Newly created numeric field*



The length and the width of the numeric field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the numeric field are controlled in the Properties window (see Fig. 4-62).

**Fig. 4-62** *Properties of a numeric field*

Properties of Numeric field	
Property	Value
Name	Nr000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	47.2826
Top	44.0217
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Horizontal alignment	Right
Length	0
Leading zeros	<input type="checkbox"/>
Rounding	No rounding
Always negative	<input type="checkbox"/>
Minimum value	
Maximum value	
Configure boxing	...
Link to object	

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for numeric fields, or have additional information pertinent to the description of the common properties.

## Stylesheet

Formatting and presentation of the numbers is controlled via the stylesheet (see section 4.7.1).

**Helpful Hint:** *When unexpected results in the way the numbers are displayed are encountered, verifying the stylesheet in the stylesheet editor may provide an explanation.*

## Horizontal alignment

The horizontal alignment default setting is **right**. This facilitates the presentation of numbers in a column.

## Length

Maximum number of characters in the field. When this number of character is reached, any further entered characters are ignored. No value or the value 0 mean that the field has no length limitation.

**Note:** *Snapform has, as almost any other software too, a technically caused limitation of the size of numbers. This limit is at 16 significant digits. Numbers with more than 16 digits are either rounded or truncated.*

*It is recommended to limit entry fields to a smaller number of significant digits, when there is a chance that too many digits will be used. It is also recommended for strings consisting of digits which are not numbers (such as serial numbers) to use text fields with selected **Only digits** option.*

When filling in the form, text that is too long is displayed in smaller font size and with condensed characters, so that it completely fits into the field. The consequence is that text can become unreadable rather quickly. It is therefore important to estimate the text size when planning the form, and to provide text fields with an appropriately sized data entry field.

## Leading zeros

When this box is checked, leading zeros are not suppressed. This is important for serial numbers or certain types of page numbers.

When the box gets checked, the value of **Length** is set to 8, if it was 0 before, and **Rounding** is set to the value **Integer**.

**Note:** *When this box gets accidentally checked, the two mentioned settings must be carefully checked and manually be set to their original values.*

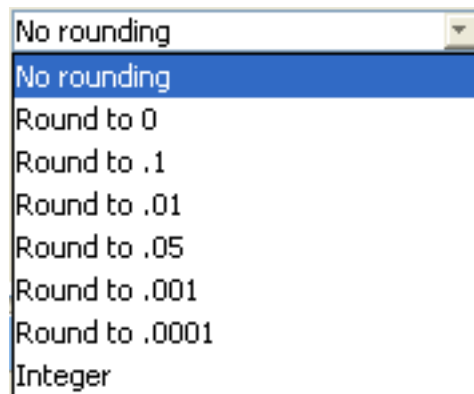


## Rounding

Rounding normally occurs as the result of a calculation, not when data is manually entered. After a calculation, there may be more digits after the decimal sign than wanted, which may disturb the display of numbers. It is therefore reasonable to show rounded values. With this property, the rounding behavior is set.

When this property is selected, a drop-down menu appears with the various options (see Fig. 4-63).

**Fig. 4-63** *The Rounding drop-down menu*



The default is **No Rounding**, which means that entered and displayed numbers are not rounded. When one of the following options **Round to 0**, **Round to .1**, **Round to .01**, **Round to .05**, **Round to .001**, **Round to .0001** is selected, the entered and displayed numbers will be rounded to the according precision. The option **Integer** means that only integers are accepted, and that entering (and displaying) of a decimal sign is not possible.

**Note:** *As calculations are done at full precision, there may be discrepancies between the displayed results and the calculations based on the rounded values. This is a consequence of error propagation, and must be taken into account for any specific application.*

*If necessary, the calculation must be adjusted in the way that numbers in the expression are rounded, using **Round (x)**, and the calculation is consistently done with rounded numbers (knowing that the result may not be exact).*

## Always negative

When this box is checked, the according number is always treated as a negative number.

## Minimum value

When this property has a value, the entered value will be validated and accepted only when it is greater than or equal to the specified minimum value. (see also the Attention line to **Maximum value** below).

## Maximum value

When this property has a value, the entered value will be validated and accepted only when it is smaller than or equal to the specified maximum value.

When the value is outside of the range specified by **Minimum value** and **Maximum value**, a message appears which indicates the accepted range, and the entered value gets discarded. An empty field (or a deleted entry) will not be validated.

**Attention:** *When a minimum or maximum value has been specified, it can no longer be deleted; it can be modified, however.*

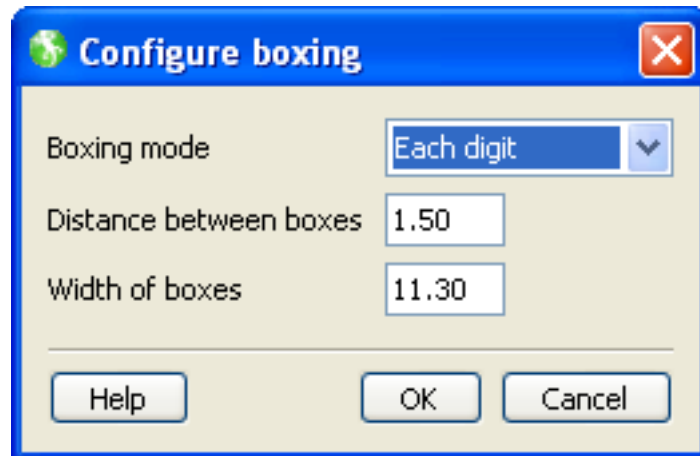
## Configure boxing

With numeric fields, various possibilities for segmenting/boxing are available. Normally, the field gets segmented into individual boxes for each character. This kind of boxed fields is also called "comb fields".

**Note:** *Note that the segmenting/boxing of fields is purely a matter of presentation. The actual field value is not modified when the field gets boxed (which becomes apparent in the way it shows when the field is active).*

Clicking on ... for this property opens the configuration window (see Fig. 4-64).

**Fig. 4-64** Configuration window for field boxing in numeric fields

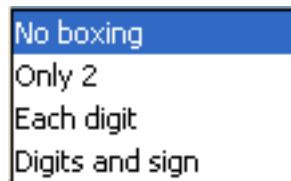


The configuration window for field boxing allows you to set the following properties:

## Boxing mode

This drop-down menu controls the boxing types of the field (see Fig. 4-65).

**Fig. 4-65** drop-down menu for field boxing types in numeric fields



## No boxing

This is the default, and the field will not be boxed. The fields for **Distance between boxes** and **Width of boxes** are not active.

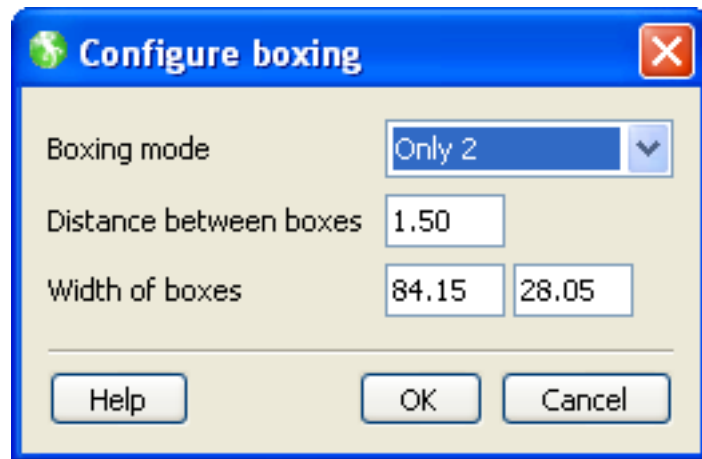
## Only 2

This boxing type separates the number in a block before the decimal sign and a block after the decimal sign. The decimal sign itself is not displayed.

**Note:** This boxing type has specifically been created for currency amounts where a non-specified number of digits before, and exactly two digits after the decimal sign are displayed. In addition, unless the rounding is set to .05, rounding to .01 is used even if another rounding is set (or none at all).

Selecting this boxing type changes the configuration window (see Fig. 4-66).

**Fig. 4-66** Configuration window for field boxing in numeric fields with the Only 2 boxing type



For further explanations, see below:

### Each digit

With this setting the field behaves like a normal comb field, where the decimal sign and the minus sign are not displayed. The window is shown in Fig. 4-64.

### Digits and sign

With this setting the field behaves like a normal comb field, where the decimal sign is not displayed. The minus sign is however displayed. The window is shown in Fig. 4-64.

### Distance between boxes

The distance between the boxes for the individual characters, measured in Points. The default value is 1.5 pt. This value applies to all boxing types.

### Width of boxes

The width of the box for the individual characters in Points. The default value is 11.3 pt. This value applies to the boxing types **Each digit** and **Digits and sign**. For the boxing type **Only 2** the width of the box before the decimal sign (default value 74.55 pt) and the width of the box after the decimal sign (default value 24.85 pt) are defined.


**Note:** *As the default value for the width of boxes is not connected to the font size defined in the stylesheet, it is important to make sure that the width of the boxes is sufficient when larger font sizes are used. As a rule of thumb use the value of the font size of the field for just numbers, and 1.2 times the value of the font size of the field for general text.*

For the boxing types **Each digit** and **Digits and sign** the width of the entry field is recalculated according to the number of characters specified in the **Length** property, and the field is scaled accordingly. If there is no length set, or the value is 0, the entry field is set to a length of 8 characters. This value is also carried forward to the **Length** property. With the boxing type **Digits and sign** an additional box is added which is reserved for the minus sign. The boxing type **Only 2** has no effect on the number of characters.

#### 4.4.2.5

### Date fields

Date fields are specialized entry fields to enter date and time. They also allow you to specify date ranges.

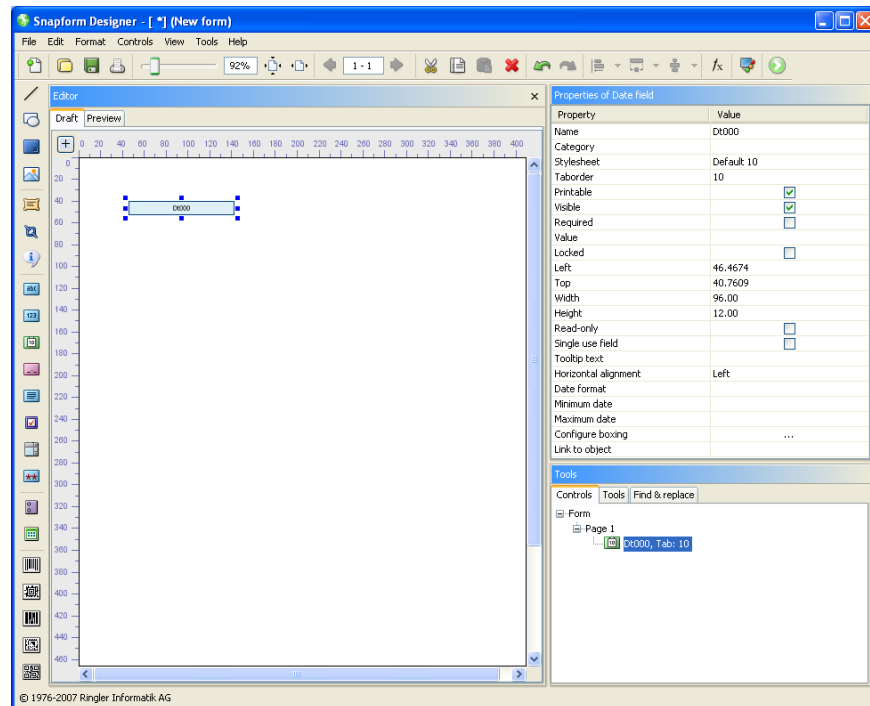
The Date field tool  creates a field which allows displaying and entering date and time. After selecting the tool, a cursor with a Date field element in default size appears (see Fig. 4-67).

**Fig. 4-67** Date field cursor



Assigned to the Date field tool is a date field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Dt** and a three-digit consecutive number, beginning with **000**. The first text field placed in a form has therefore the name **Dt000**. In the object structure it is added to the according page. In Fig. 4-68 the first date field of the form has been placed and then selected.

**Fig. 4-68** *Newly created date field*



The length and the width of the date field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the date field are controlled in the Properties window (see Fig. 4-69).

**Fig. 4-69** *Properties of a date field*

Properties of Date field	
Property	Value
Name	Dt000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	46.4674
Top	40.7609
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Horizontal alignment	Left
Date format	
Minimum date	
Maximum date	
Configure boxing	...
Link to object	

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for date fields, or have additional information pertinent to the description of the common properties.

## Stylesheet

The base format of the date entry is controlled via stylesheet (see section 4.7.1).

**Helpful Hint:** *When unexpected results occur in the way the date is displayed, verifying the stylesheet in the stylesheet editor may provide an explanation.*

## Value

**Note:** *In order for a date entered as default to be properly displayed, it must be entered in the expression editor in the format **yyyyMMddHHmm**. The presentation will then be as defined in the Date format property.*

## Date format

This property allows you to override the base format of the stylesheet, and a differing date format can be specified.

Clicking on this property opens a drop-down menu from which the available formats can be selected. It is also possible to define a custom format which will then also be available in the drop-down menu. The building blocks for the date format have the following meaning:

<b>yy</b>	Year, two digits (e.g. 08)
<b>yyyy</b>	Year, four digits (e.g. 2008)
<b>M</b>	Month, without leading zero (e.g. 5)
<b>MM</b>	Month, with leading zero (e.g. 05)
<b>d</b>	Day, without leading zero (e.g. 7)
<b>dd</b>	Day, with leading zero (e.g. 07)
<b>H</b>	Hour, without leading zero (e.g. 8)
<b>HH</b>	Hour, with leading zero (e.g. 08)
<b>mm</b>	Minute (e.g. 15)

As separator allowed are: "." (Period), "/" (Slash), " " (Space), ":" (Colon).

## Minimum date

When this property has a value, the entered date will be validated and accepted only when it is later or equal than the specified minimum date. (see also notes to **Maximum date**).



## Maximum date

When this property has a value, the entered date will be validated and accepted only when it is earlier or equal than the specified maximum date.

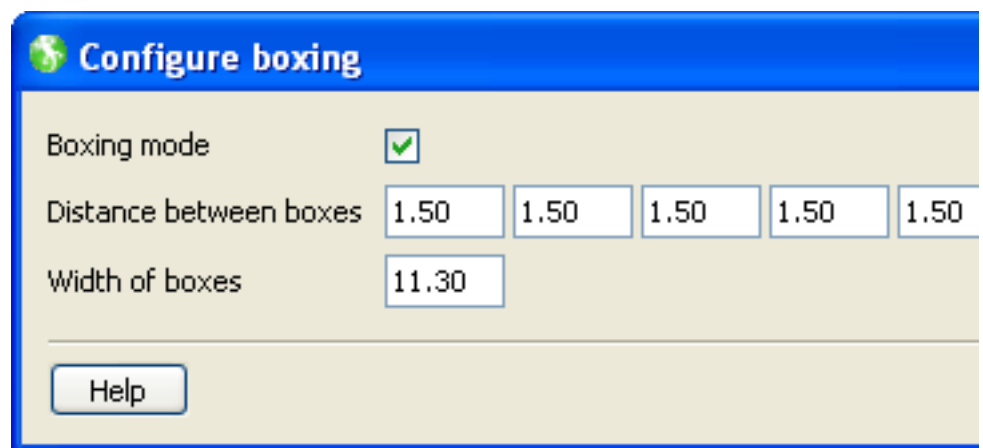
When the value is outside of the range specified by **Minimum date** and **Maximum date**, a message appears which indicates the accepted range, and the entered date gets discarded. An empty field (or a deleted entry) will not be validated.

## Configure boxing

Date fields are always boxed character by character. This kind of field is also called “comb fields”.

Clicking on ... for this property opens the configuration window (see Fig. 4-70).

**Fig.4-70** Configuration window for field boxing in date fields



The configuration window for field boxing allows you to set the following properties:

### Boxing mode

This check box activates and deactivates the field boxing. When the box is checked, the field gets boxed, when it is unchecked, the field does not get boxed.

### Distance between boxes

The distance between the boxes for the individual characters in Points. Default value is 1.5 pt. In date fields, the distance between the boxes can

be individually set. The number of fields for the spaces depends on the selected date format (for the format **d.M.yy**, 3 fields are available; for the format **dd.MM.yyyy**, there are 7 fields).

**Helpful Hint:** *It is recommended to verify the settings at a high zoom factor.*

## Width of boxes

The width of a box for the individual characters measured in Points. The default value is 11.3 pt.


### 4.4.2.6

## Masked input

Data entry fields often need to be in a specific format, in order to be valid. A simple example is a certain required date format (which is handled with a date field). Such an entry must match a “mask”. The entry field used for such purpose is the Masked input field.

Masked input fields verify the entered values according to the specified mask and accept only characters which are valid for that specific position. They may convert them accordingly. They may also contain fixed characters which are inserted automatically.

When filling out a form, and a masked input field is active, the mask is shown as default. The entry positions are marked with a place holder. When data is entered, the placeholders are replaced character by character with the entered characters.

The Masked input field tool  creates a field which allows displaying and entering of structured strings. After selecting the tool, a cursor with a Masked input field element in default size appears (see Fig. 4-71).

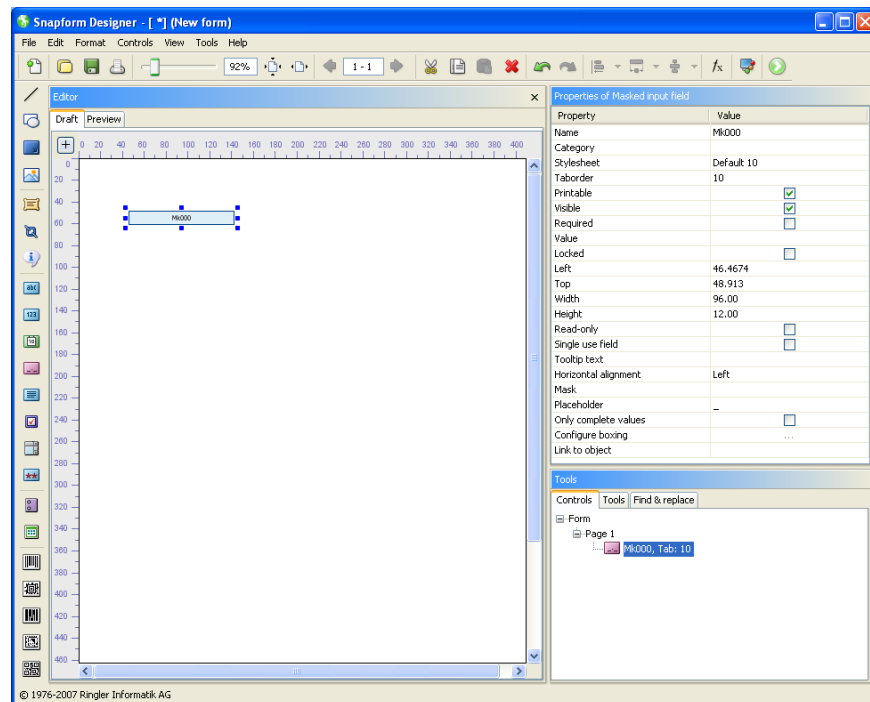
**Fig.4-71** *Masked input field cursor*



Assigned to the Masked input field tool is a masked input field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is

automatically assigned a name, consisting of the characters **Mk** and a three-digit consecutive number, beginning with **000**. The first masked input field placed in a form has therefore the name **Mk000**. In the object structure it is added to the according page. In Fig. 4-72 the first masked input field of the form has been placed and then selected.

**Fig.4-72** *Newly created masked input field*



The length and the width of the masked input field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the masked input field are controlled in the Properties window (see Fig. 4-73).

**Fig.4-73** *Properties of a masked input field*

Properties of Masked input field	
Property	Value
Name	Mk000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	46.4674
Top	48.913
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Horizontal alignment	Left
Mask	
Placeholder	-
Only complete values	<input type="checkbox"/>
Configure boxing	...
Link to object	

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for masked input fields, or have additional information pertinent to the description of the common properties.

## Stylesheet

Formatting and presentation of the entry and the appearance of the field is controlled via the stylesheet (see section 4.7.1).

## Mask

The core of the masked input field is the mask. The mask controls the structure of the value to be entered or displayed. The mask is a string which consists of specific mask characters plus other constant characters. The following mask characters are available:

- #** Number
- U** Upper case letter; lower case letters are automatically converted to upper case.
- L** Lower case letter; upper case letters are automatically converted to lower case.
- A** Letter or number
- ?** Letter (without consideration of the case)
- \*** Any character
- H** Hexadecimal character (0-9, a-f or A-F)
- '** (Apostroph), escape character; is used to “protect” a control character, if it is constant part of the mask as a literal.

Examples of masked input fields can be found in the sample document *masks.qdf*.

When a mask character is to appear literally (as itself) in a mask, it is protected with an apostrophe (the apostrophe is the Escape character).

## Placeholder

When filling out a mask input field, the mask itself is displayed. Constant characters are displayed as such, and for the characters to be entered, a placeholder is used. The character used as a placeholder is specified in this property. Normally it is the Underscore (“\_”), but any character (such as Period, Space, Bullet, etc.) can be used as a placeholder.

The mask **(0##) ### ## ##** looks in Preview mode as shown in Fig. 4-74. For a better visualization, the Period character was used as a placeholder.

**Fig.4-74** *Active masked  
input field in  
Preview mode*



## Only complete values

When this box is checked, an entry is only committed when it is complete, which means that all placeholder characters must have been replaced with an entry. An incomplete entry gets discarded.

## Configure boxing

In masked input fields the boxing occurs always for each individual character, including the constant characters of the mask. This kind of fields is also called “comb fields”.

Clicking on ... for this property opens the configuration window (see Fig. 4-75).

**Fig.4-75** *Configuration  
window for field  
boxing in masked  
input fields*



The configuration window for field boxing allows you to set the following properties:

## Boxing mode

This check box activates and deactivates the field boxing. When the box is checked, the field gets boxed, when it is unchecked, the field does not get boxed.

## Distance between boxes

Distance between the boxes for the individual characters in Points. The default value is 1.5 pt. For masked input fields, the distance between the boxes can be individually set. The number of fields for the distance values depends on the mask definition.

**Helpful Hint:** *It is recommended to verify the settings at a high zoom factor.*

## Width of boxes

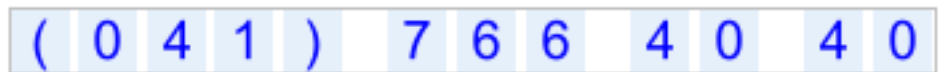
The width of a box for the individual characters measured in Points. Default value is 11.3 pt.

The mask for the example in Fig. 4-75 is **(0##)#####**. As the spaces between the boxes can be individually controlled, the mask itself does not need space characters for the formatting. The resulting field in Draft mode is shown in Fig. 4-76, and filled out in Preview mode in Fig. 4-77.

**Fig.4-76** Segmented entry field in Draft mode



**Fig.4-77** Segmented entry field in Preview mode

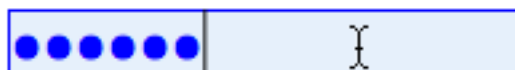


### 4.4.2.7

## Password fields

There are entries which must not be visibly displayed, such as passwords. For this kind of entry, password fields are used. Password fields are essentially text fields where the display of the entered characters is suppressed. Instead of the entered character, a big dot is shown. Fig. 4-78 shows a password field with entered data in Preview mode.

**Fig.4-78** Password field with entry in Preview mode



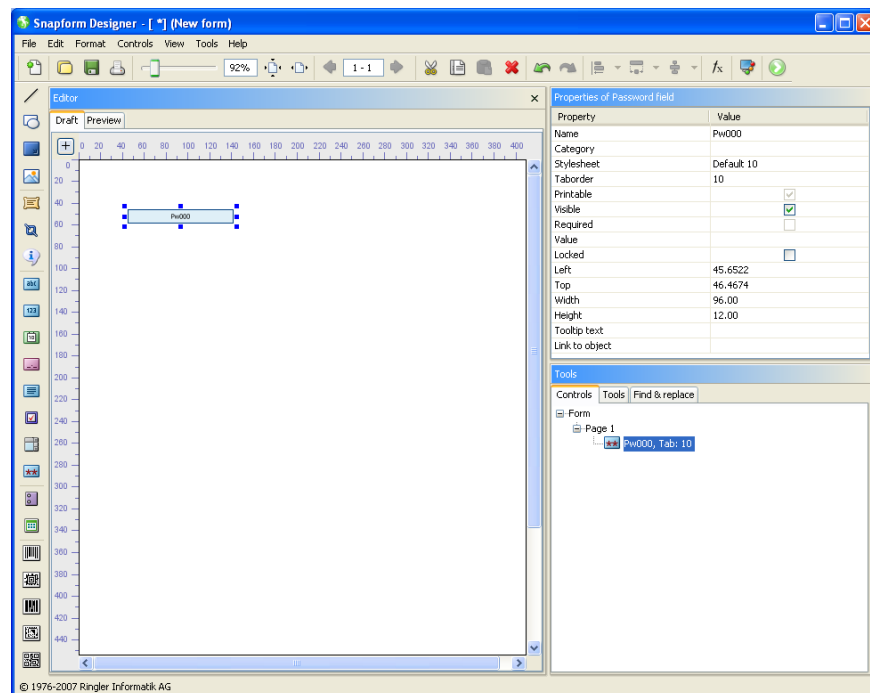
The Password field tool creates a field which suppresses the entry in the clear and displays a replacement character instead. After selecting the tool, a cursor with a Password field element in default size appears (see Fig. 4-79).

**Fig.4-79** Password field cursor



Assigned to the Password field tool is a password field, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Pw** and a three-digit consecutive number, beginning with **000**. The first password field placed in a form has therefore the name **Pw000**. In the object structure it is added to the according page. In Fig. 4-80 the first password field of the form has been placed and then selected.

**Fig. 4-80** *Newly created password field*



The length and the width of the password field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the password field are controlled in the Properties window (see Fig. 4-81).



**Fig. 4-81** *Properties of a password field*

Properties of Password field	
Property	Value
Name	Pw000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	45.6522
Top	46.4674
Width	96.00
Height	12.00
Tooltip text	
Link to object	

The common properties of entry fields are described in section 4.4.2.1. The following properties are either specific for password fields, or have additional information pertinent added to the description of the common properties.

## Stylesheet

Formatting and presentation of the entry and the appearance of the field is controlled via the stylesheet (see section 4.7.1).

## Printable

A password field is not printable, and cannot be made printable.

## Value

It is possible to specify a default value, but this does not make much sense, as it is displayed as a series of dots. However, a validation expression can be assigned to the field.

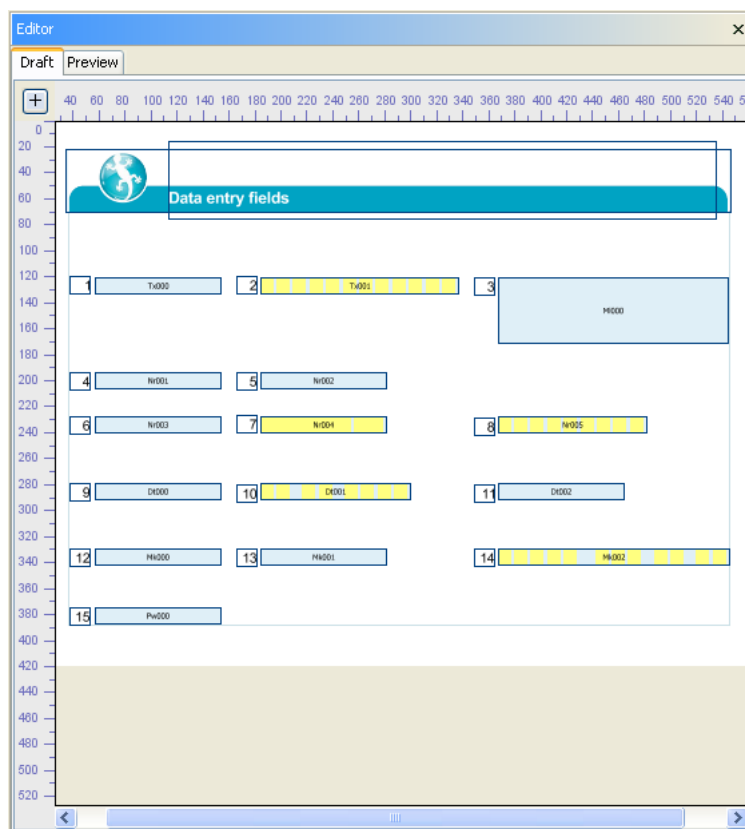
### 4.4.2.8

## Examples of entry fields

The document *entryfields.qdf* (see Fig. 4-82 (Draft mode) and Fig. 4-83 (shown in the Snapform Viewer, filled out)) contains various examples of entry fields, as described in the previous section.

After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the text.

**Fig. 4-82** Examples of entry fields of *entryfields.qdf*, in Draft mode



**Fig. 4-83** Filled out examples of Entry fields of entryfields.qdf in the Snapform Viewer

The screenshot shows a window titled 'eingabefelder.qdf \* - Snapform Viewer' with a zoom level of 150%. The form inside has a blue header with a globe icon and the title 'Data entry fields'. It contains 15 numbered examples of input fields:

- 1 Base value
- 2 Entry value
- 3 A multiline field with several entered values and lines like this.
- 4 4,352,342,345.55
- 5 45.55
- 6 00004532
- 7 34232 22
- 8 - 2 3 2 3 2 4 4
- 9 13.04.2007
- 10 2 4 1 2 2 0 0 6
- 11 12/12/2004
- 12 434-34-6432
- 13 124 ADF 25
- 14 ( 0 4 1 ) 7 6 6 4 0 4 0
- 15 .....

- 1 Tx000: Simple text field with default value
- 2 Tx001: Boxed text field
- 3 MI000: Multiline text field
- 4 Nr001: Numeric field, no rounding, no format
- 5 Nr002: Numeric field, value between 10 and 150, rounding to .05
- 6 Nr003: Numeric field with leading zeros, 8 digits
- 7 Nr004: Boxed numeric field with separated decimal sign and rounding to .001
- 8 Nr005: Boxed numeric field with Digits and sign, length 8 digits and rounding to .01
- 9 Dt000: Simple date field, format **dd.MM.yyyy**
- 10 Dt001: Boxed date field, format **dd.MM.yyyy**
- 11 Dt002: Simple date field, format **dd/MM/yyyy**; the entered date must be in the 21st century
- 12 Mk000: Masked input field, Mask **###-##-####** (Social Security Number, US)
- 13 Mk001: Masked input field, Mask **### AAA 25** (License plate of the French Departement Drôme)
- 14 Mk002: Masked input field, boxed, mask **(0##)#####** (Swiss phone number); boxing separates the number blocks; placeholder is #
- 15 Pw000: simple password field

### 4.4.3

## Check boxes and Radio buttons

A frequent element of a form is to indicate whether a certain statement is correct, or which option applies for a certain statement. In paper forms, this question is implemented with a series of boxes to check. In electronic forms, this is done with check boxes and radio buttons.

A check box is a form element where a certain statement can be marked as “applicable” or “not applicable”.

Radio buttons are a group of form elements which identify various mutually exclusive options for a certain statement, and exactly one option must be applicable.


As these definitions say, check boxes and radio buttons are closely related. This is also apparent in the way these two form elements have been implemented in Snapform.

The base element is the check box which provides the base functionality. For radio buttons, a “Container” is used in which the according number of check box fields is placed. These check boxes have now become radio buttons according to the definition above.

#### 4.4.3.1

### Check box

In Snapform, check boxes are displayed in the default view as squares, and the character for “checked” is a cross. This character is fixed, and cannot be changed. The size of the character depends on the field size. If, however, the check box checking is the result of a calculation, the cross is displayed somewhat smaller.

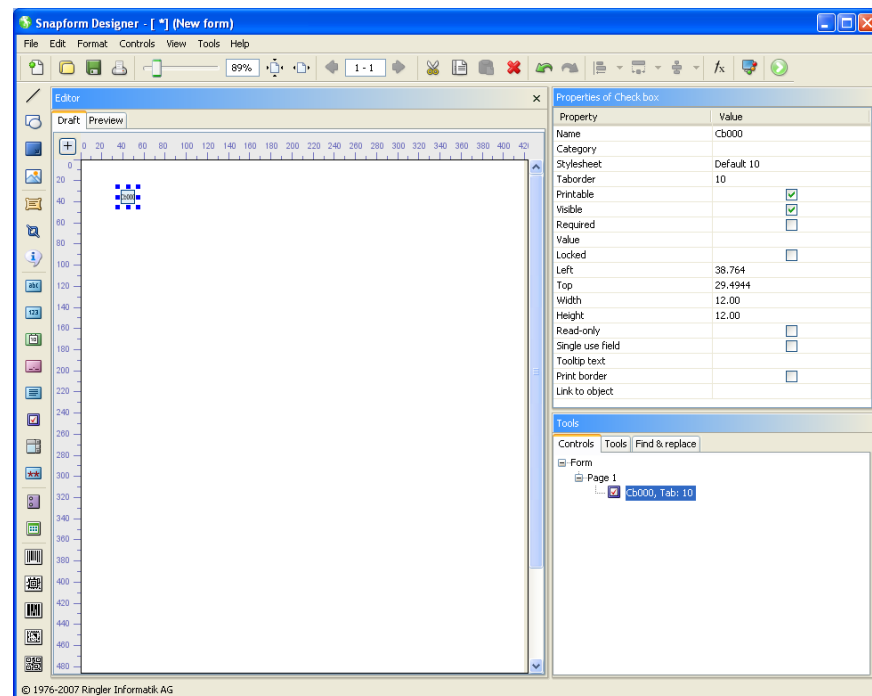
The Check box tool  creates a check box. After selecting the tool a cursor with a check box in default size appears (see Fig. 4-84).

**Fig.4-84** *Check box cursor*



Assigned to the Check box tool is a check box, 12 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Cb** and a three-digit consecutive number, beginning with **000**. The first check box placed in a form has therefore the name **Cb000**. In the object structure it is added to the according page. In Fig. 4-85 the first check box field of the form has been placed and then selected.

**Fig. 4-85** *Newly created check box*



The length and the width of the check box can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the check box are controlled in the Properties window (see Fig. 4-86).

**Fig. 4-86** *Properties of a check box*

Properties of Check box	
Property	Value
Name	Cb000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	38.764
Top	29.4944
Width	12.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Print border	<input type="checkbox"/>
Link to object	

## Name

The name of the check box in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For check boxes, the stylesheet settings in the **General settings** are of importance.

## **Taborder**

Sequence number of the element in the tab order. This value is assigned automatically, and is in the order of the field tab sequence (beginning with 10 and incremented in steps of 10). The first field of the form has tab order value 10, the second the value 20, etc.. The tab order value can be changed, so that a specific tabbing sequence can be created which has no relationship with the order the fields have been added.

## **Printable**

When the box is checked, the check box will be printed.

## **Visible**

When the box is checked, the check box is displayed on screen.

## **Required**

When this box is checked, the check box must have been checked or unchecked at least once, in order for the form to be valid (more about validity of forms in section 4.8.4.2).

## **Value**

It is possible to specify a constant value for a check box, but this does not make much sense. Instead, the expression which controls the actions happening when the box is checked or unchecked is entered here.

The expression is evaluated when the state of the box is toggled, as well as when the document opens and other expression evaluation processes are run.

## **Locked**

When this box is checked, the check box field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the check box.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the check box.

## Width

Width of the bounding box of the Check box element.

## Height

Height of the bounding box of the Check box element.

## Read-only

When this box is checked, the check box is marked as read-only, and its content cannot be manually changed. The value may still be changed as the result of an expression. A read-only check box is still part of the tab order, and its assigned expressions are evaluated according to its state.

## Single use field

There is information which may no longer be modified after initially filling out the form, and whose fields must therefore be protected. In Snapform, this protection occurs in Single use fields, which can be modified until the form is saved for the first time. After that, the field is marked as read-only.

When this box is checked, the check box can be toggled until the form is saved for the first time. After saving for the first time, the check box will be marked as read-only.

## Tooltip text

The Tooltip text is help text that appears when the mouse cursor hovers over the field for a certain time. This text is in many cases the first and simplest help to the according field. With increased requirements for accessibility, the tooltip text becomes more and more important because screen readers use this information. It is particularly important for government forms, that the form designer inserts well thought out tooltips.

Double-clicking on this property opens the **Edit text value** window (see Fig. 4-48). As indicated in the explanatory text, it is also possible to enter text in HTML form, for which the same rules apply as for labels. For further notes about tooltip text see the description in section 4.4.2.1.



**Note:** *Tooltip text should be short. Therefore it is advised to avoid entering excessive HTML formatting and inserting images.*

## Print border

When this box is checked, the border of the check box field is printed, otherwise it is not. This option is useful with forms without background image, because otherwise, it is not possible to recognize a check box on the printout.

## Link to object

In many forms applications there are references to other sections of the form, or to additional information. With the **Link to object** property it is possible to create such a reference in a Snapform form.

When this property is selected, a drop-down list containing all the elements of the form layer (all elements with a taborder value greater than 0, including Image fields), which can be selected. When the form is filled out, a green arrow mark appears at the right of the field (see also Fig. 4-52). Clicking on this icon sets the focus to that referenced field.


As described in section 4.3.6.1, it is possible to link a label with a check box, which then allows the check box to be toggled by clicking on the label.

### 4.4.3.2

## Radio buttons

In order to implement radio buttons, there must be a way to logically group check boxes, so that they will be mutually exclusive. The Snapform tool to achieve this is the Radio button container.

The Radio button group (container) is a field which makes any check box, newly created and placed within its perimeter, into a part of the set of radio buttons. In the object structure, these check boxes are one level below the other fields. The Radio button group does not quite belong to the form level because it is not part of the tab order (Taborder value -1). Outside of the Draft mode it is also invisible (in the Preview mode and the Snapform Viewer).

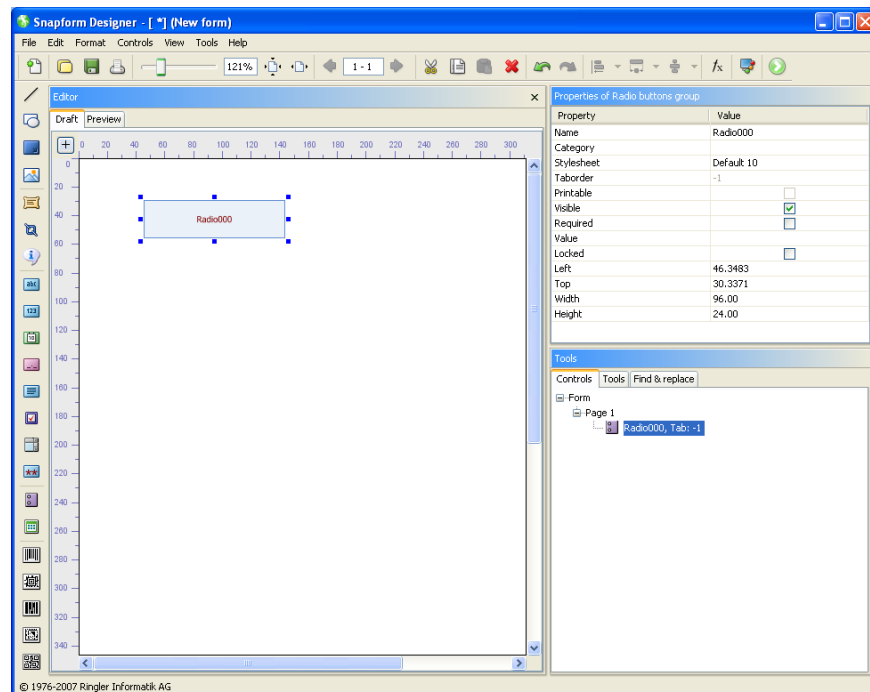
The Radio button group tool  creates a radio button container. After selecting the tool a cursor with a radio button container in default size appears (see Fig. 4-87).

**Fig.4-87** *Radio button group cursor*



Assigned to the Radio button group tool is a radio button container, 96 pt wide and 24 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Radio** and a three-digit consecutive number, beginning with **000**. The first radio button container placed in a form has therefore the name **Radio000**. In the object structure it is added to the according page. In Fig. 4-88 the first radio button container of the form has been placed and then selected.

**Fig.4-88** *Newly created radio button container*



The length and the width of the radio button container can be changed by dragging the anchor points of the bounding box. The position of the

rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the radio button container are controlled in the Properties window (see Fig. 4-89).

**Fig.4-89** *Properties of a radio button group*

Properties of Radio buttons group	
Property	Value
Name	Radio000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	46.3483
Top	30.3371
Width	96.00
Height	24.00

## Name

The name of the radio button group. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## **Stylesheet**

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used.

## **Taborder**

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence.

## **Printable**

This check box is unchecked and cannot be changed. Radio button containers are never printed.

## **Visible**

When the box is checked, the radio button container is displayed on screen.

## **Required**

When this box is checked, at least one check box belonging to the radio button group must have been checked or unchecked at least once, in order for the form to be valid (more about validity of forms in section 4.8.4.2).

## **Value**

It is possible to specify a constant value for an radio button group, but this does not make much sense. Instead, the expression which controls the actions happening when switching between the check boxes within the radio button container is entered here.

The expression is evaluated when the state of a check box in the radio button group is toggled, as well as when the document opens and other expression evaluation processes are run.

## **Locked**

When this box is checked, the radio button container and all its contained check boxes get locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive (also for the check boxes within the container). Access to these properties is only available after deactivating this check box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the radio button group.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the radio button group.

## Width

Width of the bounding box of the radio button container.

## Height

Height of the bounding box of the radio button container.

### 4.4.3.3

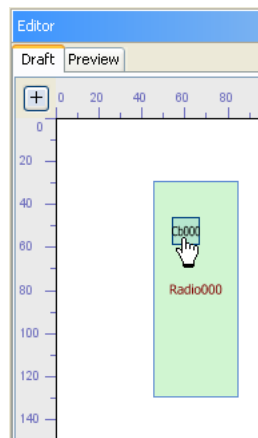
### Setting up radio buttons

The components for radio buttons are described in the two previous sections. This is how radio buttons are set up.

The first step is making the radio button container. For practical reasons, it should cover the area where the radio buttons will be located.

The second step is placing check boxes within the radio button container. These check boxes are automatically assigned to the radio button container, and turn into radio buttons. When the first check box is placed in a radio button container, it changes its color (see Fig. 4-90).

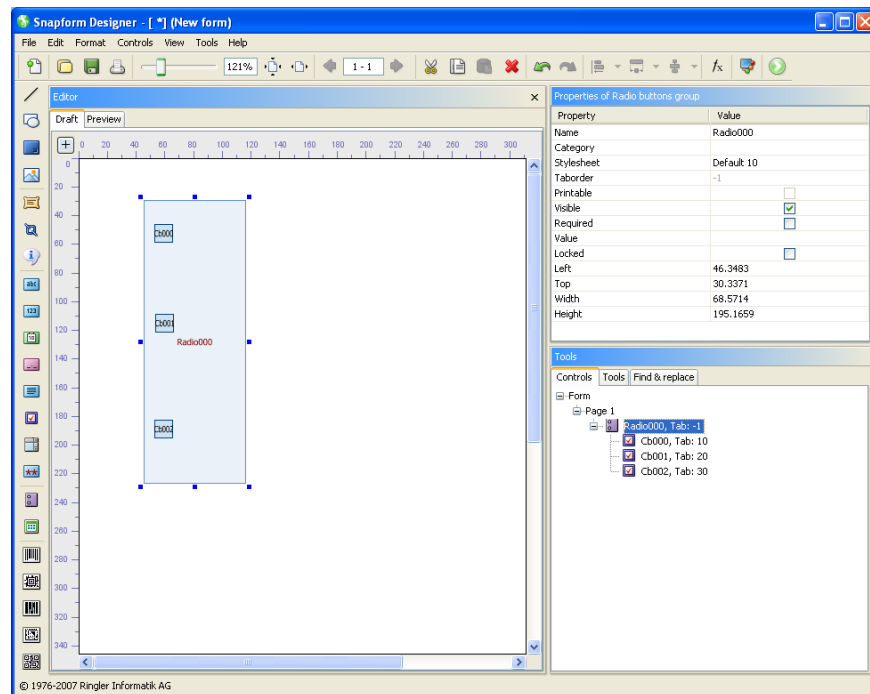
**Fig.4-90** *Placing a check box in an radio button container*



The color change is the visual indication that the check box has become part of the radio button set.

This step is now repeated until all intended options have an assigned check box (see Fig. 4-91).

**Fig.4-91** *Radio button container with placed check boxes*



The document structure shows in the example the association of the check boxes **Cb001**, **Cb002** and **Cb003** to the radio button container **Radio000**.

Check boxes which are present in the form when the radio button container is placed cannot be added to the radio button group, except when they are placed again using Cut/Paste (check the tab order in this case). Check boxes placed later on, outside of the radio button container, cannot be added to the radio button group either.

Check box fields which have been placed within a radio button container cannot be dragged outside of the container. However, it is possible to change the size of the radio button container, so that check boxes will be outside of its perimeter. These check boxes are still part of

the radio button group, and they are moved along with the container when it gets moved around on the workspace.

There are cases where the option of a check box or radio button is repeated (for example, in a form with a response page). In this situation, add the check box at the given place. If there are radio buttons, make sure that the newly placed check box will not be part of the radio button group. This new check box will be set read-only (so that it cannot be manually overridden). Let's assume, this check box needs to repeat the state of **cb001**, its value will get the expression

**= Cb001**

When checking the form in Preview mode, this box will always repeat the state of Cb001.

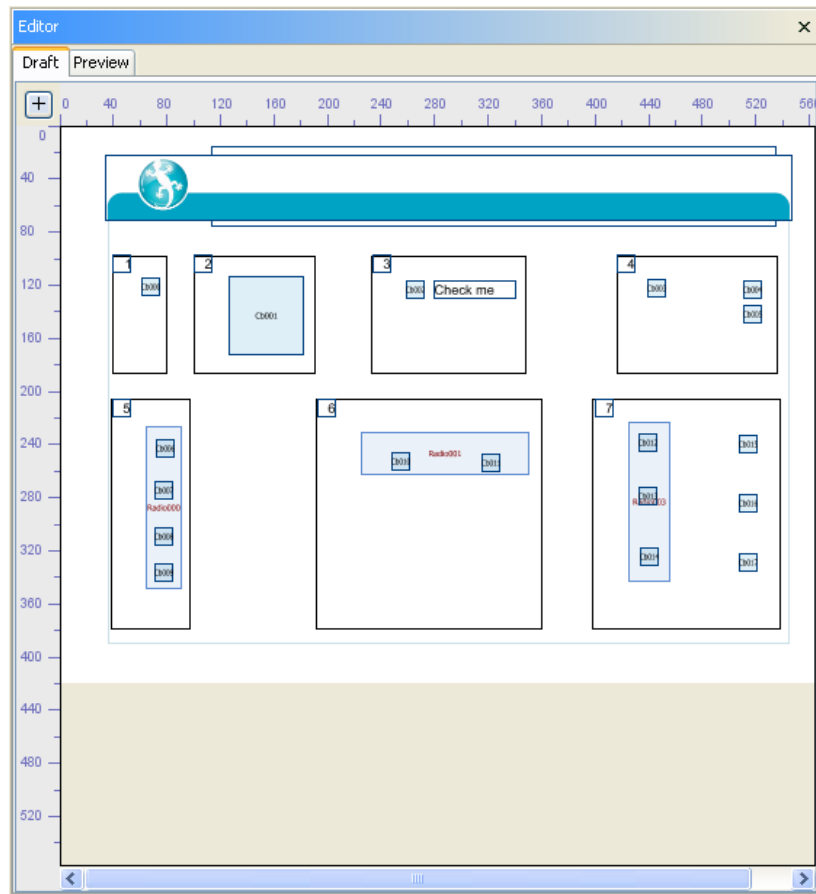
#### 4.4.3.4

### Examples of Check boxes and Radio buttons

The document *checkboxbuttons.qdf* (see Fig. 4-92 (Draft Mode) and Fig. 4-93 (shown in the Snapform Viewer, filled out)) contains various examples of check boxes and radio buttons.

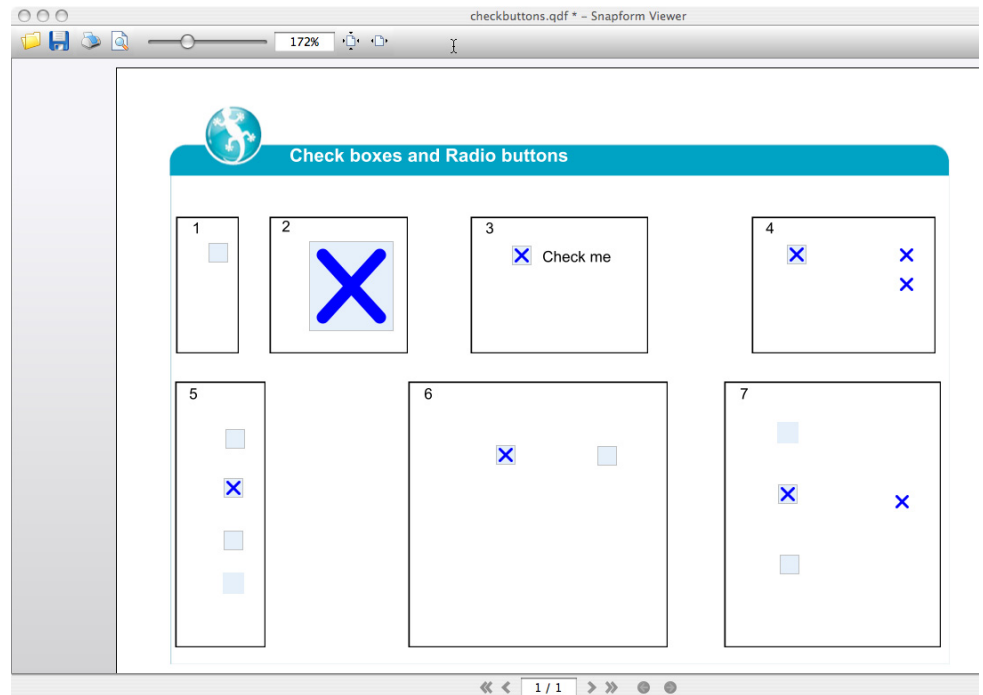
After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (as far as there is one assigned to the field).

**Fig. 4-92** Check box and  
radio button  
examples of  
checkboxbuttons.qdf,  
in Draft mode





**Fig. 4-93** *Filled out checkbox and radio button examples of checkbuttons.qdf in the Snapform Viewer*



- 1 Cb000: Simple check box in original size
- 2 Cb001: Big check box
- 3 Cb002: Check box, linked to the label Lb000
- 4 Cb003: Check box with two depending check boxes Cb004 and Cb005; the latter ones are read-only
- 5 Radio000: Radio button container with four assigned check boxes Cb006, Cb007, Cb008 and Cb009
- 6 Radio001: Radio button container with two check boxes Cb010 and Cb011; the size of the container has been changed so that the check boxes are now outside of its perimeter
- 7 Radio002: Radio button container with three check boxes Cb012, Cb013 and Cb014, and three dependent check boxes Cb015, Cb016 and Cb017; the latter are read-only and represent the state of the radio buttons

## 4.4.4

## Selection lists


A very common form element is a list of options from which the user can chose. This element has no direct equivalent in paper forms, but is extensively used in electronic forms (and in this case preferable to a big collection of radio buttons).

Selection lists have two pieces of information: the list value (face value) and the selection value (return value). The list value is displayed when the list is shown, and the selection value is displayed in the list field after it has been selected. This is also the value which can be used in further calculations.

If no selection value has been defined, the list value is used instead.

Selection lists can also be assembled using expressions.

Selection lists are implemented as Combo box fields. Combo box fields offer a selection from a static or dynamically created list of options.

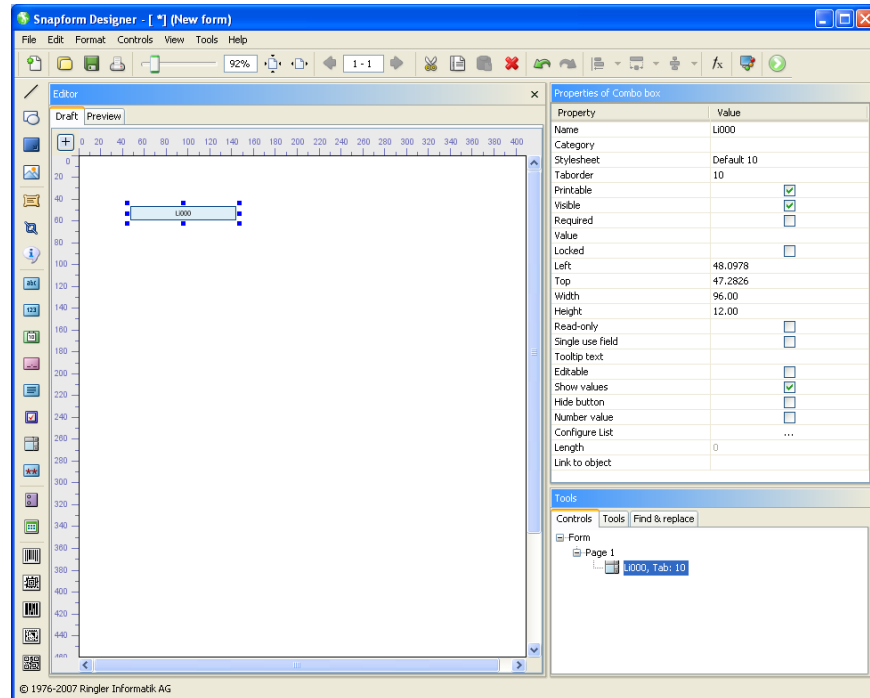
The Combo box tool  creates a field which allows to chose from a list of options. After selecting the tool, a cursor with a Combo box field in default size appears (see Fig. 4-94).

**Fig. 4-94** *Combo box field cursor*



Assigned to the Combo box tool is a combo box, 96 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Li** and a three-digit consecutive number, beginning with **000**. The first combo box placed in a form has therefore the name **Li000**. In the object structure it is added to the according page. In Fig. 4-95 the first combo box field of the form has been placed and then selected.

**Fig. 4-95** *Newly created  
combo box field*



The length and the width of the Combo box field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the Combo box field are controlled in the Properties window (see Fig. 4-96).

**Fig. 4-96** *Properties of a  
Combo box field*

Properties of Combo box	
Property	Value
Name	Li000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	48.0978
Top	47.2826
Width	96.00
Height	12.00
Read-only	<input type="checkbox"/>
Single use field	<input type="checkbox"/>
Tooltip text	
Editable	<input type="checkbox"/>
Show values	<input checked="" type="checkbox"/>
Hide button	<input type="checkbox"/>
Number value	<input type="checkbox"/>
Configure List	...
Length	0
Link to object	

## Name

The name of the Combo box field. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For combo boxes the stylesheet settings in the **General settings** are of importance.

## Taborder

Sequence number of the element in the tab order. This value is assigned automatically, and is in the order of the field tab sequence (beginning with 10 and incremented in steps of 10). The first field of the form has tab order value 10, the second the value 20, etc. The tab order value can be changed, so that a specific tabbing sequence can be created which has no relationship with the order the fields have been added.

## Printable

When the box is checked, the combo box will be printed.

## Visible

When the box is checked, the combo box is displayed on screen.

## Required

When this box is checked, a selection must have been made from the combo box, in order for the form to be valid (more about validity of forms in section 4.8.4.2).

## Value

It is possible to specify a constant value for a combo box, but this does not make much sense because it will not be displayed. Instead, the expression which controls the actions happening when a selection is made is entered here.

The expression is evaluated when the selected element of the list is changed, as well as when the document opens and other expression evaluation processes are run.

## Locked

When this box is checked, the Combo box field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the combo box.

## **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the combo box.

## **Width**

Width of the bounding box of the combo box.

## **Height**

Height of the bounding box of the combo box.

## **Read-only**

When this box is checked, the combo box is marked as read-only, and its content cannot be manually changed. The value may still be changed as the result of an expression. A read-only combo box is still part of the tab order, and its assigned expressions are evaluated according to its selected value.

## **Single use field**

There is information which may no longer be modified after initially filling out the form, and whose fields must therefore be protected. In Snapform, this protection occurs in Single use fields, which can be modified until the form is saved for the first time. After that, the field is marked as read-only.

When this box is checked, the selection of the combo box can be changed until the form is saved for the first time. After saving for the first time, the combo box will be marked as read-only.

## **Tooltip text**

The Tooltip text is help text that appears when the mouse cursor hovers over the field for a certain time. This text is in many cases the first and simplest help to the according field. With increased requirements for accessibility, the tooltip text becomes more and more important because screen readers use this information. It is particularly important

for government forms, that the form designer inserts well thought out tooltips.

Double-clicking on this property opens the **Edit text value** window (see Fig. 4-48). As indicated in the explanatory text, it is also possible to enter text in HTML form, for which the same rules apply as for labels. For further notes about tooltip text see the description in section 4.4.2.1.

**Note:** *Tooltip text should be short. Therefore it is advised to avoid entering excessive HTML formatting and inserting images.*

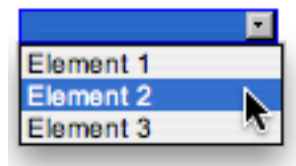
## Editable

When this box is checked, it is possible to enter an additional value to the available list. This value remains in the field until a new selection is made. This value is not added to the selection list, and when the selection is changed, it will be lost.

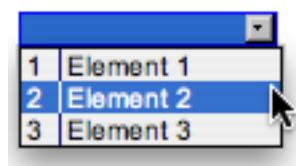
## Show values

When this box is checked, the selection value is shown with the list value when the combo box is opened (see Fig. 4-97 (box unchecked) and Fig. 4-98 (box checked)).

**Fig.4-97** Selection with **Show values** unchecked



**Fig.4-98** Selection with **Show values** checked



## Hide button

When this box is checked, the button right of the Combo box field, which is normally shown and indicates that there is a combo box, is hidden. This button will then appear only when the field is active.

## Horizontal alignment

This property defines how the entered or displayed text is horizontally aligned in the combo box.

When the property field is clicked, a drop-down menu appears with the options **left**, **center**, **right**, which places the text left-aligned, centered or right-aligned within the combo box. Default is **left**.

## Number value

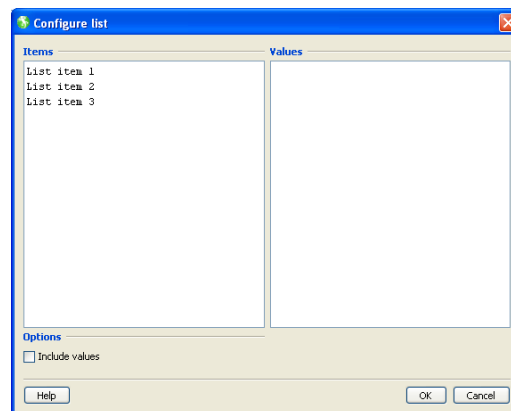
When this box is checked, the selection value must be a number.

## Configure list

In this property, the actual selection list is specified. The list value must always be entered, the selection value may be entered if requested.

Clicking on ... for this property opens the configuration window (see Fig. 4-99).

**Fig.4-99** Configuration window for selection list



The configuration window consists of two text columns and a check box in the **Options** area.

## Include values

When this box is checked, the values of the **Values** column are used for the selection list. The **Values** column also becomes accessible (when this box is unchecked, the **Values** column is locked).

## Items

The list values of the selection list items are entered into the **Items** column. Each item has to be on its own line. The length of the list is not limited.

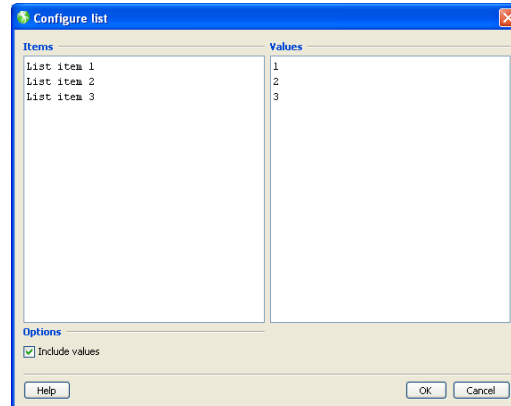


## Values

In the **Values** column, the selection values associated to the list values in the **Items** column are specified. This column is only accessible when the Include values check box is checked (see Fig. 4-100).

**Note:** *The Items and Values column do not support HTML-formatted entries.*

**Fig.4-100** Configuration fields for selection list with active Values list



## Length

This value limits the length of entered values to the according number of characters. When this value is 0, there is no length limitation.

## Link to object

In many forms applications there are references to other sections of the form, or to additional information. With the **Link to object** property it is possible to create such a reference in a Snapform form.

When this property is selected, a drop-down list containing all the elements of the form layer (all elements with a taborder value greater than 0, including Image fields), which can be selected. When the form is filled out, a green arrow mark appears at the right of the field (see also Fig. 4-52). Clicking on this icon sets the focus to that referenced field.

A selection list can also be dynamically assembled with an expression. The function used for this purpose is called **setList()** and is explained in section 4.5.2.9.

## 4.4.5


### 1D Barcodes

1D barcodes allow encoding of small amounts of data as a barcode which allows paper-based workflows to be supported electronically. The amount of encodable data is, however, small (depending on the system, up to 40 characters). This is the reason why 1D barcodes are not suitable for transporting payload data. However, 1D barcodes are ideal for encoding data about the document (identification, routing information, case identifier, etc.) to be used in archival or storage systems.

The choice of the “right” 1D barcode system depends on various aspects (such as character set, data length, readability, legal requirements or available infrastructure) and must be considered in the analysis process. In this section, there will be no in-depth description of the actual barcode systems. Instead of that, refer to the specific literature. A short description of the barcodes supported in Snapform can however be found in section 4.4.5.2.

#### 4.4.5.1

#### The Barcode tool

The Barcode tool  creates a field in which a 1D barcode can be displayed. After selecting the tool, a cursor with a Barcode element in default size appears (see Fig. 4-101).

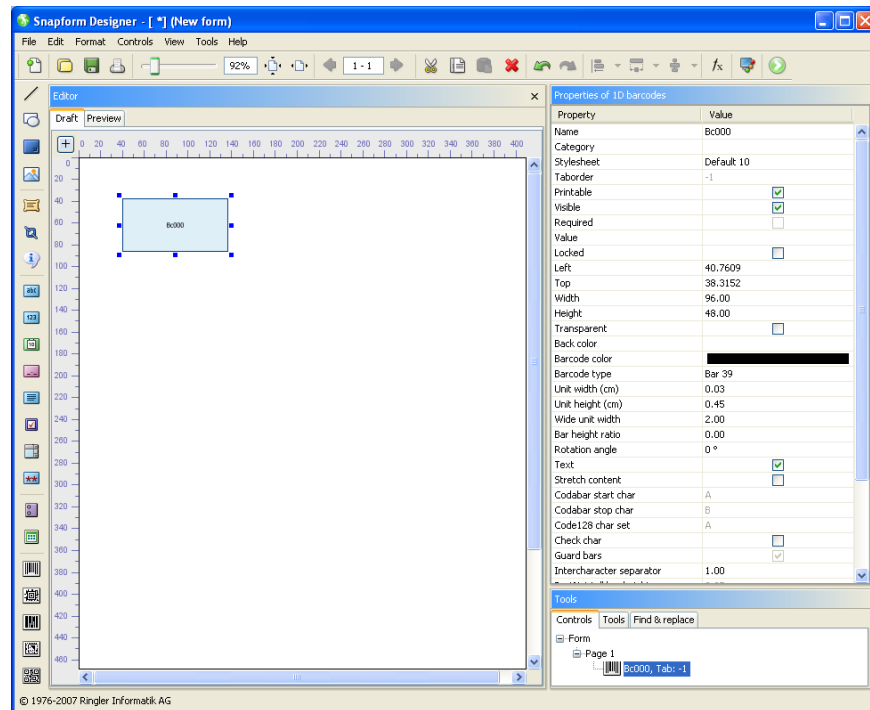
**Fig.4-101** 1D Barcode cursor



Assigned to the Barcode tool is an 1D barcode field, 96 pt wide and 48 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Bc** and a

three-digit consecutive number, beginning with **000**. The first barcode field placed in a form has therefore the name **Bc000**. In the object structure it is added to the according page. In Fig. 4-102 the first barcode field of the form has been placed and then selected.

**Fig.4-102** Newly created barcode field




The length and the width of the barcode field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the barcode field are controlled in the properties window (see Fig. 4-103).

**Note:** *The barcode field has several properties which are relevant only for specific barcode systems. These properties are shown, but are only active when the according barcode type has been selected.*

**Fig. 4-103** *Properties of a  
barcode field*



Property	Value	
Name	Bc000	▲
Category		
Stylesheet	Default 10	
Taborder	-1	
Printable	<input checked="" type="checkbox"/>	
Visible	<input checked="" type="checkbox"/>	
Required	<input type="checkbox"/>	
Value		
Locked	<input type="checkbox"/>	
Left	40.7609	
Top	38.3152	
Width	96.00	
Height	48.00	☰
Transparent	<input type="checkbox"/>	
Back color		
Barcode color		
Barcode type	Bar 39	
Unit width (cm)	0.03	
Unit height (cm)	0.45	
Wide unit width	2.00	
Bar height ratio	0.00	
Rotation angle	0 °	
Text	<input checked="" type="checkbox"/>	
Stretch content	<input type="checkbox"/>	
Codabar start char	A	
Codabar stop char	B	
Code128 char set	A	
Check char	<input type="checkbox"/>	
Guard bars	<input checked="" type="checkbox"/>	
Intercharacter separator	1.00	
PostNet tall bar height	0.25	
PostNet short bar height	0.125	
Supplement bar height factor	0.80	
Supplement distance (cm)	0.50	
2 digit supplement	<input type="checkbox"/>	
5 digit supplement	<input type="checkbox"/>	
Supplement		
UPCE encoding system	1	▼

<b>Name</b>	The name of the barcode field in the document. This name can be changed.
<b>Category</b>	<p>Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).</p> <p>The categories are selected by clicking on the according check box. With <b>OK</b> the selection becomes valid.</p>
<b>Stylesheet</b>	Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For barcode fields only the Border settings in the General settings tab are used.
<b>Taborder</b>	Sequence number of the element in the tab order. This value is <b>-1</b> and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the barcode is not an active element.
<b>Printable</b>	When the box is checked, the barcode field will be printed.
<b>Visible</b>	When the box is checked, the barcode field is displayed on screen.
<b>Required</b>	This property is always deactivated in barcode fields, and cannot be changed.
<b>Value</b>	<p>Clicking in the property field opens the expression editor for entering an expression.</p> <p>Barcode fields may contain either a constant value (for example to identify a document) or the result of an expression (more frequently).</p> <p><b>Note:</b> <i>Self-checking digits of the barcode are automatically calculated and must therefore not be taken into account in an expression. The same applies</i></p>

*for the Start and Stop characters in the Codabar symbology which are specified in the according barcode field property.*

## **Locked**

When this box is checked, the barcode field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the barcode field.

## **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the barcode field.

## **Width**

Width of the barcode field in Points.

**Note:** *The actual width of the barcode depends on the actual number of characters and may therefore be considerably wider than the field itself. The symbol is created starting from the upper left corner of the field. It is therefore important to make sure there is sufficient blank space towards the right of the field.*

## **Height**

Height of the barcode field in Points.

## **Transparent**

The background of the barcode is set to transparent (no covering background color) with this check box. This property is unchecked by default.

## **Back color**

Background color of the barcode field. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.



The background color is only applied when the **Transparent** check box is unchecked.

## Barcode color

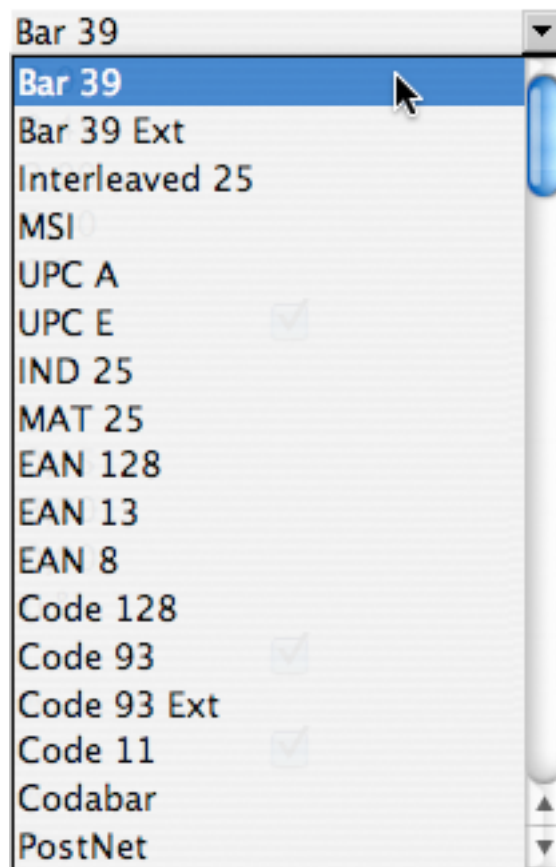
Color of the barcode. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

**Note:** *Standards for the respective barcodes may contain requirements concerning the colors to be used, and should be referred to. Also note that only the full colors Black, Cyan, Magenta and Yellow are printed as full color (unless specific spot colors are used), and that good printing quality can be achieved only with these colors at 100% saturation, as there will not be screened bars.*

## Barcode type

Type of the barcode symbol. Clicking on this property opens a drop-down menu with a list of the available barcode systems (see Fig. 4-104).

**Fig. 4-104** *List of the available 1D barcodes*



Depending on the selection of this property, further properties are activated or deactivated (see explanation of the according properties).

The following barcode symbologies are supported (for a more extensive description see section 4.4.5.2):

**Bar 39:** Code 39

**Bar 39 Ext:** Code 39 Extended

**Interleaved 25:** Interleaved Code 2 of 5

**MSI**

**UPC A**

**UPC E**

**IND 25:** Industrial Code 2 of 5, Standard Code 2 of 5

**MAT 25:** MAT25

**EAN 128**

**EAN 13**

**EAN 8**

**Code 128**

**Code 93**

**Code 93 Ext:** Code 93 Extended

**Code 11**

**Codabar**

**PostNet**

### **Unit width (cm)**

Width of the unit bar of the barcode in centimeters. This dimension influences the length of the resulting symbol. When specifying this

dimension, the expected printer's resolution must be taken into account.

**Helpful Hint:** *To calculate the approximate size in Points, multiply this value by 200 and then reduce it by 10%.*

### Unit height (cm)

Height of the unit bar of the barcode in centimeters. This dimension determines directly the absolute height of the resulting code. When its value is **0**, the height of the symbol is controlled with the **Bar height ratio** property, which sets it in relationship with the symbol width.

**Helpful Hint:** *To calculate the approximate size in Points multiply this value by 200 and then reduce it by 10%.*

### Wide unit width

1D barcodes have narrow and wide unit bars. This property of the barcode is the ratio of the widths of the wide and the narrow unit bar. This value influences the length of the resulting symbol.

Typical values are 2 and 3, whereas for many barcode symbologies 3 is required in the standard. 3 is preferable when the unit width is small. This can improve the readability of the code.

### Bar height ratio

For some barcode systems, (primarily the ones with a fixed data length) the proportions of the symbol are specified in the standard. This property calculates the height of the bars from the width of the symbol.

### Rotation angle

The barcode can be rotated by multiples of 90°. Clicking on this property opens a drop-down menu with the selection of the rotation angles **0°**, **90°**, **180°**, **270°**. The default value is 0°.

**Note:** *The direction in which the barcode grows is the same as the rotation angle.*

### Text

When this box is checked, the contents of the barcode is also displayed as text. Self-checking characters are shown, but auxiliary characters, such as Start and Stop characters are not shown. All characters relevant

to the data are shown, which are necessary when the barcode information is entered manually.

## Stretch content

When this box is checked, the symbol is scaled so that it completely fills the barcode field. This property is not active by default.

**Attention:** *Scaling a symbol to the field's dimension may lead to bar widths which are beyond the specification, and the symbol may be unreadable. This property must be used with great care, and understanding of the barcode used in the application.*

## Codabar Start char and Codabar Stop char

The Codabar code has **Start** and **Stop** characters which may occur only at the beginning and the end of the code (see description in section 4.4.5.2). These characters are named **A**, **B**, **C** and **D**, and may be used for payload purposes to some extent. **Start** and **Stop** character may be the same or different.

These two properties are only active when the **Codabar** code type has been selected. When this property is selected, a drop-down menu opens with the selection **A**, **B**, **C**, **D**. Default values are **Start** character **A** and **Stop** character **B**.

## Code 128 Char set

The Code 128 code (as well as the technically identical EAN-128 code) uses three different character sets, **A**, **B** and **C** (see description in section 4.4.5.2).

With this property, the character set used at the beginning of the code is specified. A change of character set within the code is marked automatically.

This property is only active when **Code 128** or **EAN-128** are active. When it is selected, a drop-down menu opens with the selection **A**, **B**, **C**. Default is character set **A**.

## Check char

Various barcode symbologies allow the use of self-checking characters which improves the reading reliability.

When this box is checked, a check character is inserted which is also shown in the clear. The calculation method of this check character's value depends on the selected barcode type.

**Note:** *For the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E), the check character is part of the data, and this property only has an effect when the value is shorter by 1 digit (12 digits for EAN-13, 11 digits for UPC-A and UPC-E, 7 digits for EAN-8).*

*The Codabar symbology does not have check characters.*

*In the PostNet code, Code 128 and EAN-128 as well as Code 11, the check character is a fixed component of the symbol. In these cases, this property creates an additional check character which can be evaluated by the post-processing application. The built-in check character is calculated over the complete length of data.*

## Guard bars

With the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E) it is possible to frame ("guard") the text line with extended bars at the beginning and the end, as well as between the manufacturer code and the product code.

When this box is checked, the symbol is shown with the extended bars.

This property is only active when the code type **EAN-13**, **EAN-8**, **UPC-A**, or **UPC-E** has been selected. With the other code types, it is ignored. For application in the retail business, it should always be selected.

## Intercharacter separator

Several barcode symbologies are classified as "discrete codes". In these symbologies, the equivalents of characters may be present on their own, without being part of a greater entity. In these symbologies, an empty space between the character symbols can be defined, whose task it is to create a distance between the characters. This empty space is defined as Intercharacter spaceholder and it is specified as a multiple of the unit width.

This property is only active when the code type **Code 11**, **Codabar**, **Bar 39** or **Bar 39 Ext** has been selected. With the other code types, it is ignored.

### PostNet tall bar height and PostNet short bar height

The encoding in the PostNet code is done via different bar heights. These two properties specify the absolute dimensions of the bars in centimeters.

The ratio between tall and short bar length as well as the absolute sizes are defined in the PostNet specification

The default values 0.125 and 0.25 cm follow the specification.

These properties are only active when the code type **PostNet** has been selected. With the other code types, they are ignored.

### Supplement bar height factor

With the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E) for printed products (books, periodicals), it is possible to add 2- or 5-digit supplemental codes. These supplemental codes are lower than the main symbol, and the clear text line is above their symbol.

This property specifies the relative height of the supplemental code. A typical value is 0.8.

This property is only active when the code type **EAN-13**, **EAN-8**, **UPC-A**, or **UPC-E** has been selected. With the other code types, it is ignored.

### Supplement distance (cm)

With the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E) for printed products (books, periodicals), it is possible to add 2- or 5-digit supplemental codes. These supplemental codes are lower than the main symbol, and the clear text line is above their symbol.

This property specifies the distance of the supplemental code symbol from the main symbol, measured in centimeters.

This property is only active when the code type **EAN-13**, **EAN-8**, **UPC-A**, or **UPC-E** has been selected. With the other code types, it is ignored.

## Supplement

With the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E) for printed products (books, periodicals), it is possible to add 2- or 5-digit supplemental codes. These supplemental codes are lower than the main symbol, and the clear text line is above their symbol.

This property defines the text to be displayed as supplemental code. In order to display the supplemental code, the according option (**2 digit supplement** or **5 digit supplement**) must also be selected. The supplemental symbol is displayed only if the value consists of 2 or 5 digits.

This property is only active when the code type **EAN-13**, **EAN-8**, **UPC-A**, or **UPC-E** has been selected. With the other code types, it is ignored.

### 2 digit supplement and 5 digit supplement

With the retail business codes (EAN-13, EAN-8, UPC-A, UPC-E) for printed products (books, periodicals), it is possible to add 2- or 5-digit supplemental codes. These supplemental codes are lower than the main symbol, and the clear text line is above their symbol.

When one of this boxes is checked, a supplemental code symbol consisting of 2 or 5 digits is added to the main symbol. The value of this supplement is specified in the **Supplement** property.

This property is only active when the code type **EAN-13**, **EAN-8**, **UPC-A**, or **UPC-E** has been selected. With the other code types, it is ignored.

### UPCE encoding system

The UPC-E code specification contain different encoding systems. The default value is **1**.

This property is only active when the code type **UPC-E** has been selected. With the other code types, it is ignored.

## 4.4.5.2

### 1D barcodes supported in Snapform

Snapform supports the 1D barcodes listed below. This description shall be a short help for choosing the suitable symbology; however referring

to the standards and technical descriptions of the according code will be necessary in many situations.

## Bar 39

This is Code 39, also known as 3 of 9 or USD-3. A character is represented through 9 bars where 3 are wide. This symbology is said to be the oldest alphanumeric barcode. Code 39 is a discrete symbology with variable length. It is to some extent self-checking, as a simple printing error cannot change a symbol into another valid symbol.

Valid characters: **0123456789 [Space] ABCDEFGHIJKLMNOPQRSTUVWXYZ - . \$ / + %**

Application: industry, health, military

A check character modulo 43 can be specified as an option.

## Bar 39 Ext

This is the Code 39 extended, or Full-ASCII Code 39. It is based on the Code 39, but allows displaying all 7 bit ASCII character, by combining characters of the Code 39 character set.

Valid characters: **0123456789 [Space] ABCDEFGHIJKLMNOPQRSTUVWXYZ ! # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } plus non-displayable control characters**

Application: industry, health, military

A check character modulo 43 can be specified as an option.

## IND 25

This is the Industrial 2 of 5 or Standard 2 of 5 code, a low-density numeric code. A character is represented with 5 black bars 2 of which are wide. The white space has no information content and its purpose is to separate the black bars. This is one of the oldest numeric codes. The 2 of 5 code is a discrete symbology with variable length. It does not have self-checking mechanisms built in. However, a check digit modulo 10 can be defined on application level.

Valid characters: **0123456789**



Application: industry

This code should not be used for new applications anymore, because of its very low density. Instead, if a 2 of 5 type of code must be used at all, the Interleaved 2 of 5 code should be chosen.

## Interleaved 25

This is the Interleaved 2 of 5 code, a medium-density numeric code. It is based on the Standard 2 of 5, but also uses the white space between the bars. A character is represented with black bars or white separators, two of them wide. The Interleaved 2 of 5 code is a discrete symbology with variable length. An even number of digits must always be defined; if the number of digits in the payload is odd, a leading 0 has to be added. The code does not have self-checking mechanisms. However, a check digit modulo 10 can be defined on application level.

Valid characters: **0123456789**

Application: logistics

This code is about one third shorter than a comparable Standard 2 of 5 code.

## MAT 25

This is a very rare code for which no further description could be found.

## MSI

This is the MSI or Modified Plessey code, a low-density numeric code. A character is represented with a sequence of bars and spaces, corresponding to its binary value. The MSI code is a continuous symbology with variable length. It has several self-checking options.

- Simple check digit modulo 10
- Check digit modulo 10 applied to a code already being checked with a check digit modulo 10
- Combination of check digit modulo 11 and check digit modulo 10.

Valid characters: **0123456789**

Application: libraries, retail business logistics

This code is considered obsolete and is no longer supported in modern scanning devices.

## **EAN 13 and UPC-A**

These two codes are the so-called “big” retail business codes, as they are used in retail business all over the world. They are numeric codes with a pre-defined length (UPC-A 12 characters, EAN 13 with 13 characters). The structure is identical: 1 (UPC-A) or 2 (EAN-13) digits country/type code, followed by 5 digits manufacturer code, then 5 digits product code and 1 check digit. In certain cases, the country code can consist of 3 digits, which leads to a 4 digit manufacturer code. The EAN-13 representation of an UPC-A code has a leading 0. Because of the encoding, this will lead to identical symbols, so that a product labeled with UPC-A can be read correctly with a scanner designed for EAN-13.

Valid characters: **0123456789**

Application: retail business

The EAN-13 code, using a specific “country code” is also used to represent ISBN and ISSN numbers.

For books and periodicals, the EAN-13 or UPC-A code can be expanded with a two- or five-digit supplemental code. The former is used for periodicals to encode the issue (month, week, etc.) and the latter is used to indicate the recommended minimum retail price. These supplemental codes are numeric as well.

## **EAN 8 and UPC-E**

These two codes are the so-called “small” retail business codes. With small packaging, it may be difficult to place a complete UPC-A or EAN-13 code. For this reason, a similarly structured smaller code has been developed. The UPC-E code is a reduced to 6 digits version of the 10 digits of the UPC-A code.

**Note:** *As the UPC-E code is a “reduced” version of the UPC-A code, a valid and reducible 12 digit UPC-A number must be entered as value.*

The EAN-8 code is an independent number which is centrally registered.

Valid characters: **0123456789**

Application: retail business

Similar to the EAN-13 and UPC-A code, the EAN-8 and UPC-E code can be extended with a two- or five-digit supplement (see above).

## Code 128 and EAN-128

These two codes are technically identical. The Code 128 is a high-density alphanumeric code. Code 128 is a continuous symbology with variable length. The code contains a check sum modulo 103 which is calculated automatically and not displayed.

The EAN-128 (also called UCC 128) is technically spoken Code 128, but consists of logical blocks containing relevant information for the logistics (such as weight and dimensions of the container). These blocks start with a so-called Application Identifier which is followed by a specific number of characters. The Application Identifiers are defined in the standards.

It can represent all 127 7-bit ASCII characters. Three character sets are available. Switching between them within the code is possible after specific control characters. The character sets consist of the following characters:

- Character set A: numbers, upper case letters, control and special characters
- Character set B: numbers, upper and lower case letters
- Character set C: double digits

Valid characters: **0123456789 [Space] ABCDEFGHIJKLMNOPQRSTUVWXYZ !#\$%&'()\*+,-./:;<=>?@[\\]^\_` abcdefghijklmnopqrstuvwxyz{|}** plus non-displayable control characters

Application: logistics (particularly as EAN-128) and many other applications

## Code 93 and Code 93 Ext

This is an expansion and extension of Code 39 which allows you to represent the complete 7-bit ASCII character set (with Code 93 extended). These codes are, opposed to Code 39, continuous, which results in a somewhat higher density. The code contains check sums which are not displayed in the clear text line.

Valid characters: **0123456789 [Space] ABCDEFGHIJKLMNOPQRSTUVWXYZ - . \$ / + %**

Additionally in Code 93 Ext: **! # & ' ( ) \* , : ; < = > ? @ [ \ ] ^ \_ ` a b c d e f g h i j k l m n o p q r s t u v w x y z { | } ~** plus non-displayable control characters

Application: industry

## Code 11

This is medium-density numeric code. Code 11 is a discrete symbology with variable length. The code is not very secure, because printing errors can create valid characters. However, it contains one (for payloads up to 10 characters) or two (for more than 10 characters) check sums which are not displayed in the clear text line.

Valid characters: **0123456789 -**

Application: telecommunication in North America

## Codabar

This is a medium-density quasi-numeric code. Codabar is a discrete, self-checking symbology of variable length. The code always begins with a Start character (**A, B, C, D**), and ends with a Stop character (also **A, B, C, D**). Start and Stop characters may be used to hold some information. The individual characters begin and end with a black bar, and must therefore be separated with a narrow separator.

Valid characters: **0123456789 - \$ : / . +** plus Start and Stop character

Application: logistics, photo labs

## Postcode

This is the PostNet code, developed by the US Post Office for sorting mail. PostNet is a numeric symbology with variable length (in the

original application, it was limited to 5, 9, or 11 characters payload). In the PostNet code, the data is encoded via the height of the bars (which allows for a bad printing quality (dot-matrix printers)). The code contains a check sum modulo 10 which is not displayed in the clear. For dimensions and placement of the PostNet code refer to the requirements by the USPS.

Valid characters: **0123456789**

Application: mail sorting in the US

### 4.4.5.3

### 1D barcode examples

The following examples show the various 1D barcodes with various options. Some of the options are available for several different codes, but are shown only with one code as an example.

The text fields below the according code symbol allow entering your own text. Be aware of the allowed characters for the codes, however.

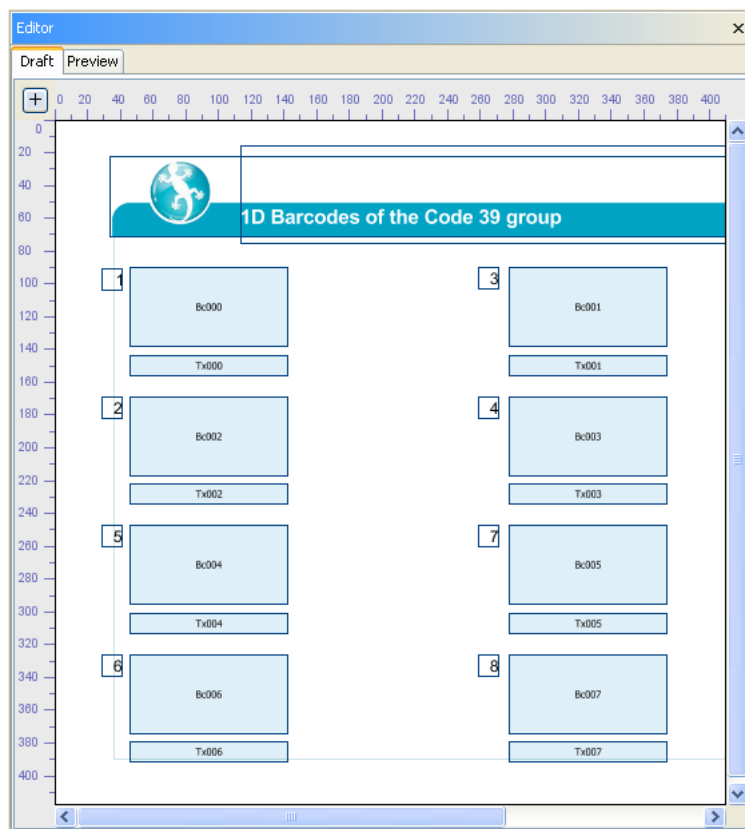
**Note:** *It is possible that the barcodes are not displayed completely in the screen shots. This effect has technical reasons, and by increasing the zoom factor, the bars appear correctly. For a proper representation of the barcodes, it is recommended that you print out the documents.*

*In the examples, dummy data is used. It is therefore possible that the actual values are illegal for the respective barcode, and that there may be errors when scanning the codes.*

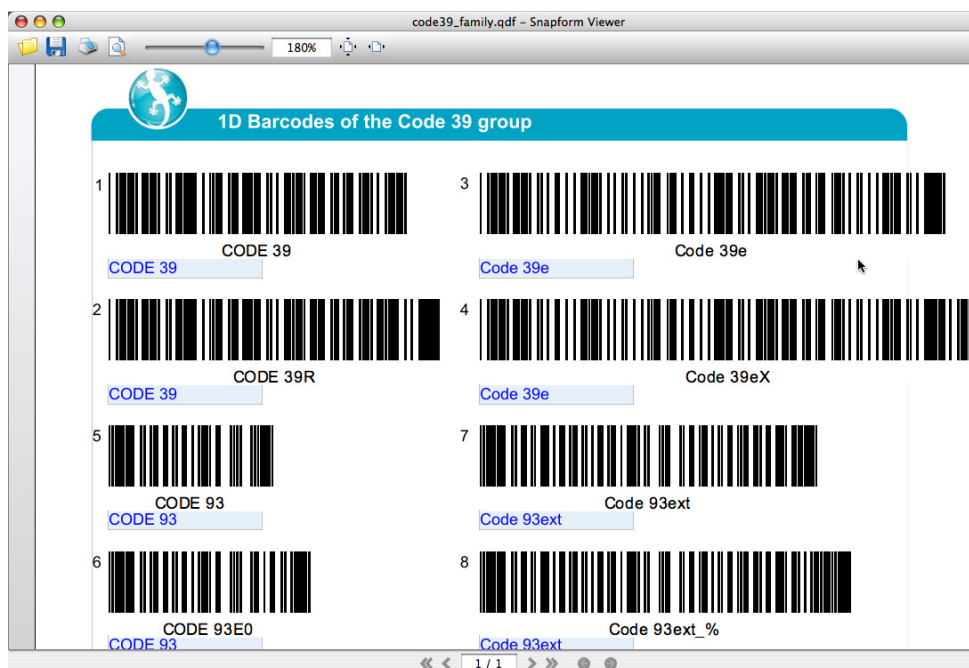
The document *code39\_family.qdf* (see Fig. 4-105 (Draft mode) and Fig. 4-106 (viewed in the Snapform Viewer)) shows various 1D barcode examples from the Code 39 group (Code 39, Code 39 Extended, Code 93, Code 93 Extended).

After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig. 4-105** Examples from  
*code39\_family.qdf*  
in Draft mode



**Fig. 4-106** Examples from  
*code39\_family.qdf*  
in the Snapform viewer

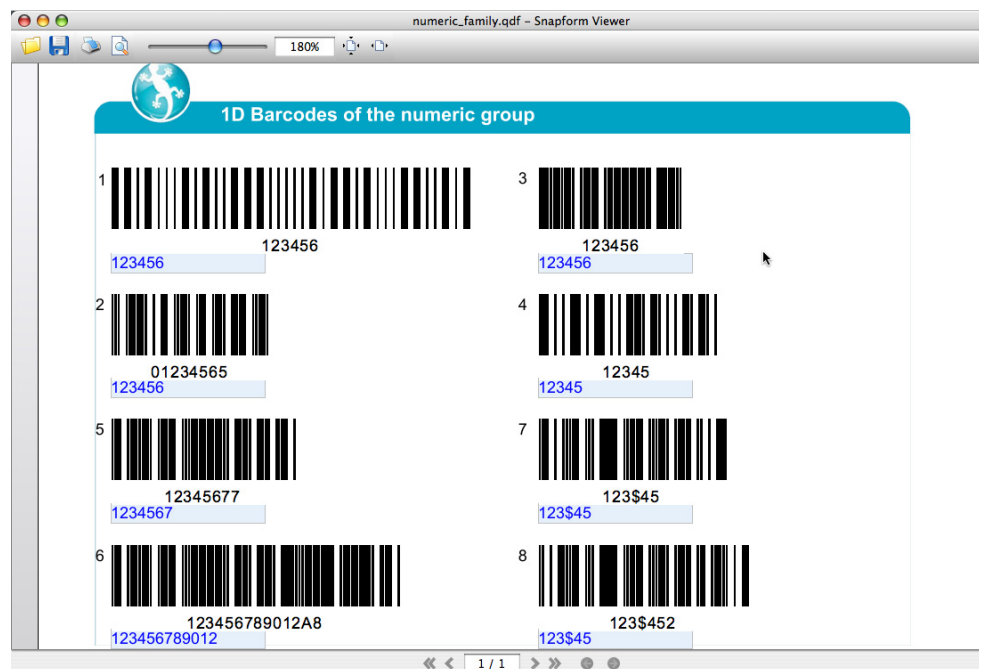


- 1 Bc000: Code 39, text **CODE 39**, without check digit
- 2 Bc002: Code 39, text **CODE 39**, with check digit
- 3 Bc001: Code 39 Extended, text **CODE 39e**, without check digit
- 4 Bc003: Code 39 Extended, text **CODE 39e**, with check digit
- 5 Bc004: Code 93, text **CODE 93**, without check digit
- 6 Bc006: Code 93, text **CODE 93**, with check digit
- 7 Bc005: Code 93 Extended, text **CODE 93ext**, without check digit
- 8 Bc007: Code 93 Extended, text **CODE 93ext**, with check digit

The document *numeric\_family.qdf* (see Fig. 4-107 (viewed in the Snapform Viewer)) shows various examples of numeric 1D barcodes .

**Note:** *The Draft mode view is identical to Fig. 4-105.*

**Fig.4-107** Examples from *numeric\_family.qdf* in the Snapform Viewer

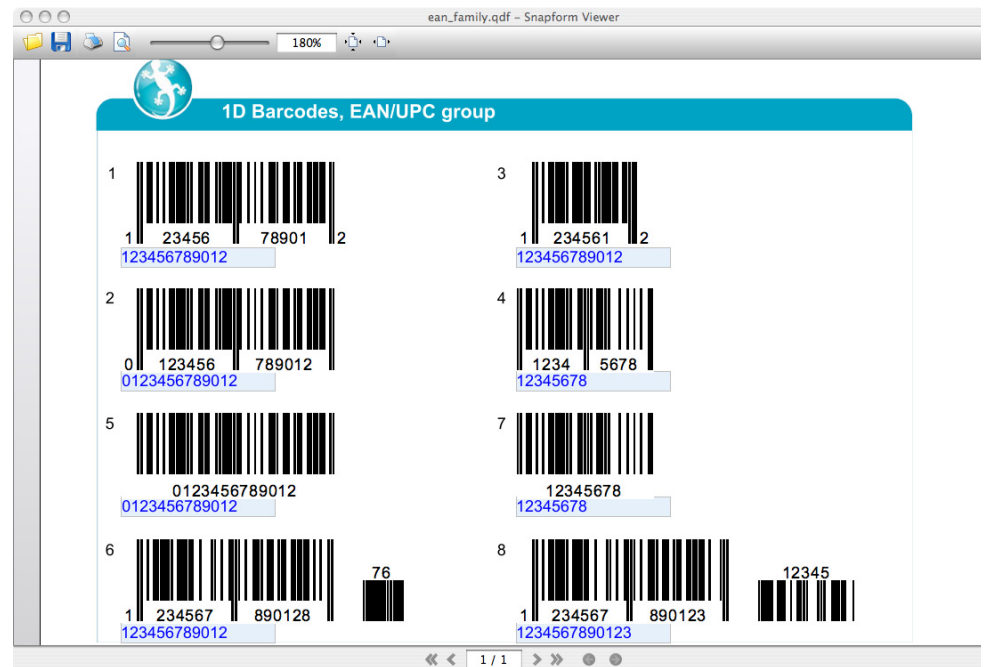


- 1 Bc000: 2 of 5 Code, text **123456**, without check digit
- 2 Bc002: Interleaved 2 of 5 Code, text **123456**, with check digit
- 3 Bc001: MAT 25 Code, text **123456**, without check digit
- 4 Bc003: MSI Code, text **123456**, without check digit
- 5 Bc004: Code 11, short text **1234567**, with check digit
- 6 Bc006: Code 11, long text **123456789012**, with check digit
- 7 Bc005: Codabar, text **123\$45**, without check digit, Start **A**, Stop **B**
- 8 Bc007: Codabar, text **123\$45**, with check digit, Start **C**, Stop **C**

The document *ean\_family.qdf* (see Fig. 4-108 (shown in the Snapform Viewer) shows different examples of the retail business barcodes (EAN-8, EAN-13, UPC-A, UPC-E).

**Note:** *The Draft mode view is identical to Fig. 4-105.*

**Fig.4-108** Examples from *ean\_family.qdf* in the Snapform Viewer



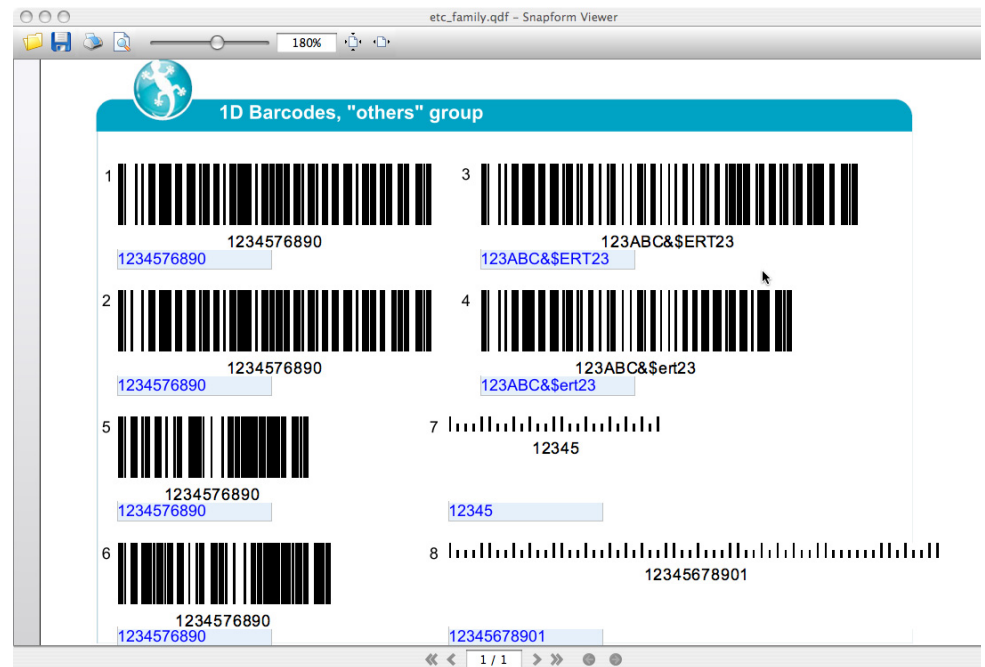
- 1 Bc000: UPC-A Code, text **123456789012**, with check digit
- 2 Bc002: EAN-13 Code, text **0123456789012**, with check digit
- 3 Bc001: UPC-E Code, text **123456789012**, with check digit; the reduction of the number of digits occurs automatically
- 4 Bc003: EAN-8 Code, text **12345678**, with check digit
- 5 Bc004: EAN-13 Code, text **0123456789012**, with check digit, without guard bars
- 6 Bc006: EAN-13 Code, text **123456789012**, with check digit, with 2-digit supplement code **76**
- 7 Bc005: EAN-8 Code, text **12345678**, with check digit, without guard bars
- 8 Bc007: EAN-13 Code, text **1234567890123**, with check digit, with 5-digit supplement code **12345**



The document *etc\_family.qdf* (see Fig. 4-109 (shown in the Snapform Viewer) shows different examples of other barcodes (Code 128, EAN-128, PostNet).

**Note:** *The Draft mode view is identical to Fig. 4-105.*

**Fig. 4-109** Examples from *etc\_family.qdf* in the Snapform Viewer

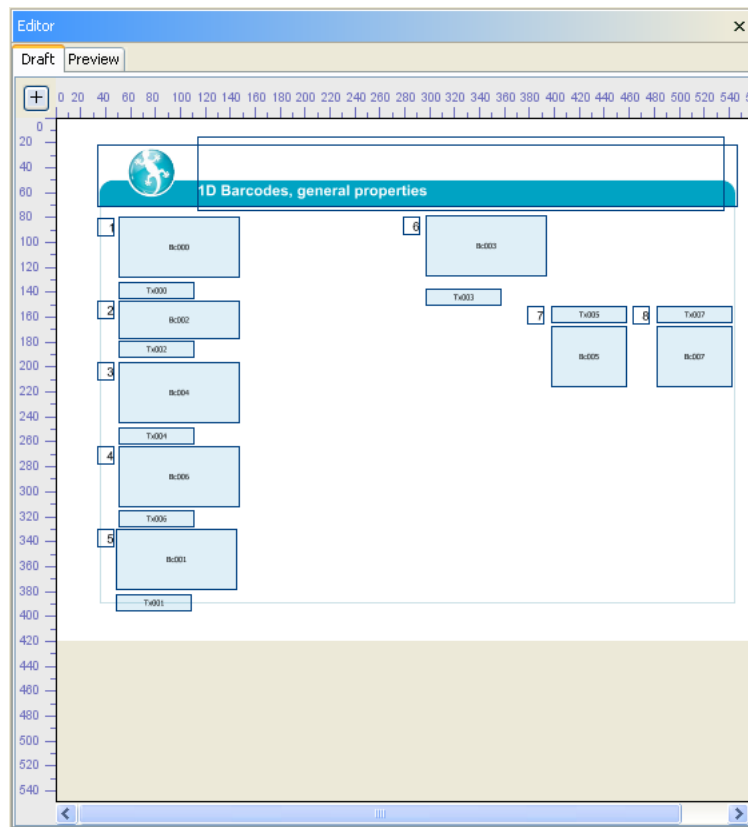


- 1 Bc000: Code 128, text **1234567890**, with check digit, Char set A
- 2 Bc002: Code 128, text **1234567890**, with check digit, char set B
- 3 Bc001: Code 128, text **123ABC&\$ERT23**, with check digit; starting char set A
- 4 Bc003: Code 128, text **123ABC&\$ert23**, with check digit, starting char set A
- 5 Bc004: Code 128, text **1234567890**, with check digit, char set C
- 6 Bc006: EAN-128 Code, text **1234567890**, with check digit, char set C
- 7 Bc005: PostNet Code, text **12345**, with check digit
- 8 Bc007: PostNet Code, text **12345678901**, with check digit

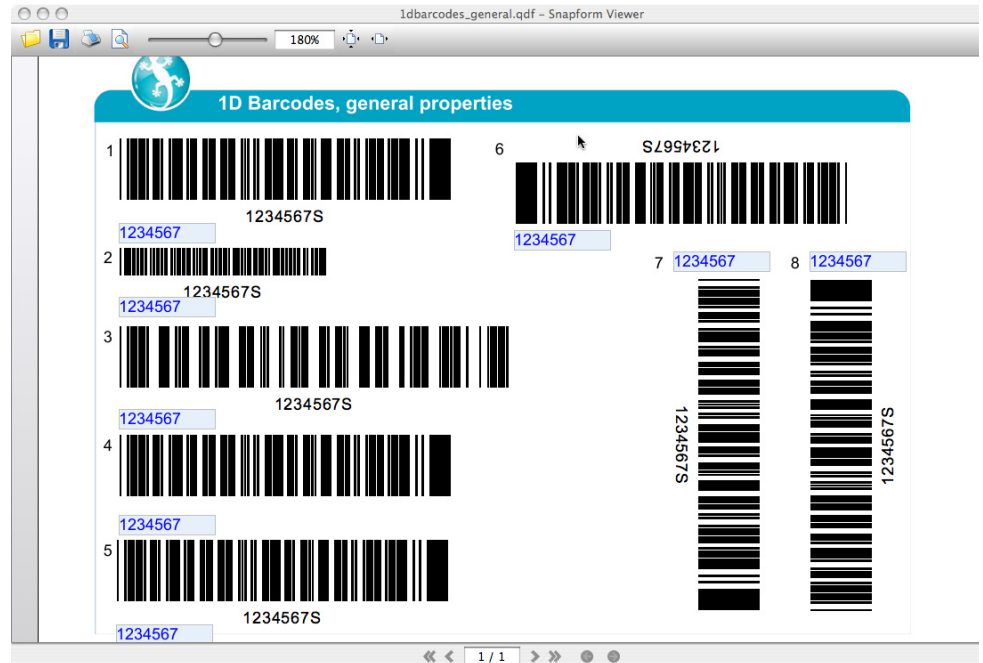
The document *1dbarcodes\_general.qdf* (see Fig. 4-110 (Draft mode) and Fig. 4-111 (shown in the Snapform Viewer)) shows various properties of 1D barcodes. The examples use Code 39 whose value is always the same.

After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig. 4-110** Examples from *1dbarcodes\_general.qdf*, in Draft mode



**Fig. 4-111** Examples from  
*1dbarcodes\_general.qdf*, in the  
*Snapform Viewer*



- 1 Bc000: Code 39, text **1234567**, with check digit, Unit width 0.04 cm, Height 1 cm, Wide unit width 3 narrow unit widths, Intercharacter separator 2 narrow unit widths
- 2 Bc002: Code 39, text **1234567**, with check digit, Unit width 0.03 cm, Height 0.45 cm, Wide unit width 2 narrow unit widths, Intercharacter separator 1 narrow unit width (these parameters are the default settings of the Snapform Designer)
- 3 Bc004: Code 39, text **1234567**, with check digit, Unit width 0.04 cm, Height 1 cm, Wide unit width 3 narrow unit widths, Intercharacter separator 5 narrow unit widths
- 4 Bc006: Code 39, text **1234567**, with check digit, same as item 1, but without text
- 5 Bc001: Code 93, text **1234567**, with check digit, Rotation angle 0°
- 6 Bc003: Code 93, text **1234567**, with check digit, Rotation angle 180°
- 7 Bc005: Code 93, text **1234567**, with check digit, Rotation angle 90°
- 8 Bc007: Code 93, text **1234567**, with check digit, Rotation angle 270°

## 4.4.6

## 2D Barcodes

2D barcodes are barcode symbologies which encode information not only in one dimension, but in two. This increases the capacity of the symbol dramatically compared to 1D barcodes: up to 2400 bytes of binary data per symbol, which (with according compression) can translate to up to 4000 characters. This allows you to encode the entire data content of a form into a small area, and it also allows you to build electronic workflows with paper-based processes.

In addition, there are barcode systems which allow you to chain symbols to increase the encodable amount of data.

For the choice of the barcode system, the circumstances of the actual application must be taken into account even more than for 1D barcodes. A discussion of the relevant points for the choice is beyond the scope of this manual. We refer to the appropriate literature and barcode system vendor support.

### 4.4.6.1

### 2D barcodes in general

When using 2D barcodes in Snapform, the following comments may be helpful.

In 2D barcodes, the symbol consists of small elements, called modules. Modules are normally squares (length ratio 1:1). Some symbologies allow other length ratios as well, which can be set.

Because of the larger amount of data and the different reading mechanisms, simple check digits are not sufficient for ensuring the data integrity. Instead of that, error correction mechanisms are built in, which allow you to retrieve the data even after reading errors occurred, or the symbol has been damaged. A higher level of security (a higher level of error correction) reduces the amount of data because the information for the error correction is contained in additional data. With an accordingly high level of error correction built in, up to half of the symbol can be damaged, and the data can still be retrieved.

The smallest information unit is the so-called code word. A code word frequently corresponds to an 8-bit byte. The data encoding occurs in Snapform by the application, and does therefore not need to be taken into further account. However, it is important for estimating the available amount of data (and therefore also for the required and available amount of data for the error correction), to know the structure of the data, in order to reduce overhead (control characters etc.).

Snapform supports four widely used 2D barcode systems: Aztec, PDF-417, Datamatrix and QRCode. As these codes are based on different concepts, it has been decided to create an individual tool for each of these four systems (instead of one "2D barcode" tool). These tools and their barcodes are explained in the following sections.

#### 4.4.6.2


#### Aztec barcode

The Aztec barcode is a square-shaped high-density matrix symbology. Its data capacity is 3832 numeric characters, 3067 alphanumeric characters, or 1914 bytes binary data. First samples of this code reminded people of Central-American pyramids, and that led to its name.

The encoded data is organized as "layers" (square rings) around a central symbol. This configuration leads to 32 different discrete symbol sizes, between 15 x 15 modules and 151 x 151 modules. Layers are always topped off with error correction information.

The central symbol is a prominent feature and can be easily recognized by the image processor. Additional modules at the corners of the central symbol specify the orientation, which allows reading the symbol from any direction. As the code is generated from the inside out, no silence area (white space around the symbol) is necessary. Even if the outermost layer cannot be reliably read, the error correction still assures reliable decoding of the data.

Characters are encoded according to ASCII (for values 0 to 127) and according to ISO 8859-1, Latin Alphabet 1 (values 128 to 255). This allows encoding text in most western world languages.

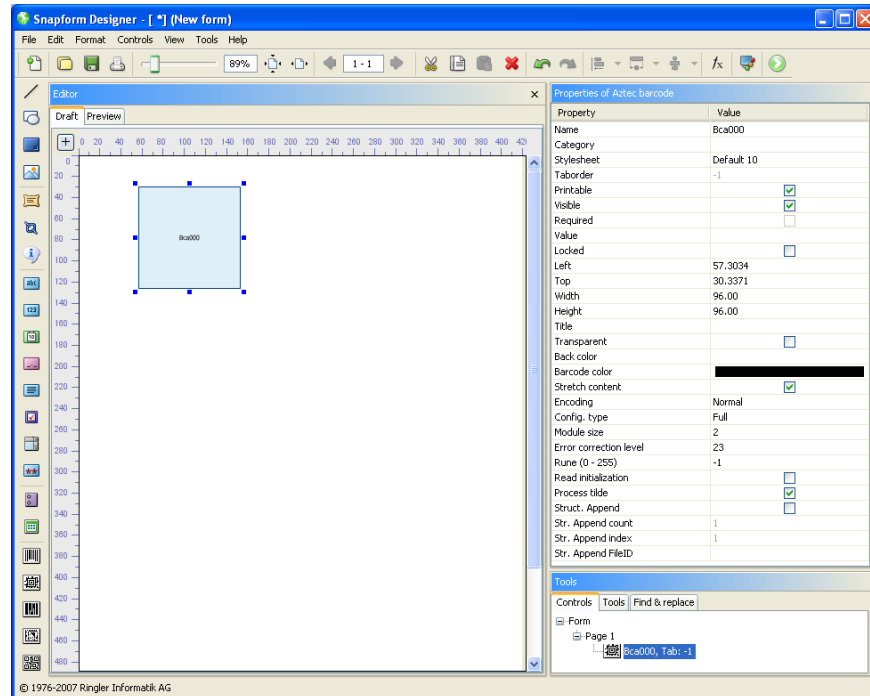
The Aztec barcode tool  creates a field which allows displaying an Aztec 2D barcode. After selecting the tool, a cursor with a 2D barcode field in default size appears (see Fig. 4-112).

**Fig.4-112** *Aztec barcode tool cursor*



Assigned to the Aztec barcode tool is a 2D barcode field for Aztec code, 96 pt wide and 96 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Bca** and a three-digit consecutive number, beginning with **000**. The first Aztec barcode field placed in a form has therefore the name **Bca000**. In the object structure it is added to the according page. In Fig. 4-113 the first Aztec barcode field of the form has been placed and then selected.


**Fig. 4-113** Newly created  
Aztec barcode field



The length and the width of the Aztec barcode field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the Aztec barcode field are controlled in the Properties window (see Fig. 4-114).

**Fig. 4-114** *Properties of an  
Aztec barcode field*

Properties of Aztec barcode	
Property	Value
Name	Bca000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	57.3034
Top	30.3371
Width	96.00
Height	96.00
Title	
Transparent	<input type="checkbox"/>
Back color	
Barcode color	
Stretch content	<input checked="" type="checkbox"/>
Encoding	Normal
Config. type	Full
Module size	2
Error correction level	23
Rune (0 - 255)	-1
Read initialization	<input type="checkbox"/>
Process tilde	<input checked="" type="checkbox"/>
Struct. Append	<input type="checkbox"/>
Str. Append count	1
Str. Append index	1
Str. Append FileID	

## Name

The name of the Aztec barcode field in the document. This name can be changed.



## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used.

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the barcode is not an active element.

## Printable

When the box is checked, the Aztec barcode field will be printed.

## Visible

When the box is checked, the Aztec barcode field is displayed on screen.

## Required

This property is always deactivated in Aztec barcode fields, and cannot be changed.

## Value

In this property of the Aztec barcode field, the text to be encoded is entered. Clicking on this property field opens the expression editor. In most of the cases, the value is the result of a calculation.

## Locked

When this box is checked, the Aztec barcode field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Aztec barcode.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Aztec barcode field.

## Width

Width of the bounding box of the Aztec barcode field.

## Height

Height of the bounding box of the Aztec barcode field.

## Title

This is a line of text which is placed above the Aztec barcode field. This title may, for example, contain information for the operator who manually scans in the code.

The title may be either simple text, or entered in HTML form. It is of fixed font size (10 pt) and its length is limited to the width of the Aztec barcode field.

## Transparent

The background of the Aztec barcode is set to transparent (no covering background color) with this check box. This property is unchecked by default.

## Back color

Background color of the Aztec barcode field. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

The background color is only applied when the **Transparent** check box is unchecked.

## Barcode color

Color of the Aztec barcode. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

**Note:** *Standards for the Aztec barcode may contain requirements concerning the colors to be used, and should be referred to. Also note that only the full colors Black, Cyan, Magenta and Yellow are printed as full color (unless specific spot colors are used), and that good printing quality can be*

*achieved only with these colors at 100% saturation, as there will not be screened modules.*

## Stretch content

This check box determines whether the symbol is scaled to the size of the Aztec barcode field. When the box is checked, the symbol is scaled to fit into the field perimeter. The value for the Module size property is in this case overridden. When the box is unchecked, the symbol is displayed using the specified module size, and it is centered in the field.

**Attention:** *When this property is active, the symbol may be scaled down so far with greater amount of data that it cannot be printed out properly anymore, and the resulting module size will be too small.*

**Note:** *When this property is not active, the symbol may grow beyond the field's size and cover other elements of the document.*

## Encoding

This property specifies in which way the data stream to be displayed in the symbol will be encoded. When it is selected, a drop-down menu with the options **Standard** and **Binary** opens. **Standard** is the default setting. The setting **Standard** optimizes the encoding of character codes below 128 (ASCII). On the other hand, the encoding of character codes above 128 is not very efficient. In the **Binary** setting, the character codes are not optimized. This setting should only be used if a considerable amount of characters with code above 127 is expected, because this setting substantially reduces the capacity of the code.

## Config. type

With a small amount of data (up to 53 bytes), a compact form of the Aztec barcode can be used, which has a smaller center symbol and may have up to 4 data layers. Such a symbol is smaller and should be processed faster.

When this property is selected, a drop-down menu opens with the three options **Automatic**, **Full**, **Compact**. **Automatic** means that the symbol is automatically switched to the compact version when the amount of data is small enough, and switched to the standard version when the

amount of data increases. **Full** means that the symbol is always displayed in the larger standard form, even with small amount of data. **Compact** means that the symbol is always displayed in the compact form.

**Attention:** *When the option **Compact** is selected and the amount of data gets too big, the symbol is displayed as a grey area.*

## Module size

This is the size of a module (the smallest unit of the Aztec barcode). Measuring unit is Point, and the default value is 2. This value should not be much smaller, as otherwise, there may be problems in printouts.

**Note:** *When the **Stretch content** property has been selected, the **Module size** value is overridden.*

## Error correction level

This is the minimum part of error correction information encoded in the symbol, in percent of the total amount of information. The default value is 23%. This value is sufficient for most applications, and allows the correct data retrieval even if up to a quarter of the symbol cannot be read correctly. Applicable values range from 5% to 95%.

## Rune

It is possible to define a series of specific symbols which have a certain meaning, defined on application level. Such a symbol contains a control sequence and an order number between 0 and 255. This kind of symbol is called "rune".

Default value for this property is **-1**, which has the meaning of "deactivated". Applicable values are integers between **0** and **255**. When an applicable value has been chosen, it will be encoded, and any values from the **Value** property will be ignored. In addition, the compact version of the symbol is automatically used (even if the Configuration type property has been set to **Full**).

## Read initialization

When this box is checked, the Aztec barcode symbol will contain the initialization flag for the reading unit (the scanner), which resets some

parameters within the reading unit. This property is deactivated as default.

## Process tilde

In order to represent control characters which cannot be entered via keyboard, the Tilde character (~) is used. This property specifies whether the Tilde character is regarded as a control character.

When this box is checked, the Tilde character is interpreted as a control character. This has the following implications which must be taken into consideration when the encoding data stream is calculated:

- ~~ is replaced with the actual tilde character (therefore double the Tilde character if one must get into the data stream).
- ~dxxx Is replaced with the character whose ASCII code is xxx (which means that, for example, ~d065 will be replaced with A). This representation is important when control characters in the lowest ASCII range are encoded.
- ~F will be replaced with the FNC1 character. This is a special character which can be at the very beginning of an Aztec barcode datastream. This character can only be placed at the first position of the datastream.
- ~Exxxxxx is replaced with the so-called Extended Interpretation Channel flag with number xxxxxx, a control character which can be evaluated on application level.

This property is deactivated in the default setting.

## Struct. Append

The Aztec barcode allows to chain up to 26 symbols, and achieve an accordingly higher data capacity. When reading chained symbols, the reading unit stores the data and assembles it. Only when all symbols have been read, the data will be passed on. This appears to subsequent processes, as if a huge symbol had been read.

When this box is checked, the according symbol is part of a structurally chained group. Otherwise, the symbol is self-contained.

When this property is selected, the properties **Str. Append count** and **Str. Append index** must have a value; they are also activated in the Properties window.

### Str. Append count

This property indicates how many symbols there are in the structurally chained group to which the symbol belongs.

Valid values are between **2** and **26**. The value **1** means that the group consists of one single symbol, and therefore, the symbol is self-contained.

This property is normally deactivated. It gets activated when the **Struct. Append** property is activated.

### Str. Append index

This property represents the index number of the symbol in the structurally chained group.

Valid values are between **1** and **26**.

This property is normally deactivated. It gets activated when the **Struct. Append** property is activated.

### Str. Append FileID

It is possible to specify in advance the file name for the retrieved chained data. That file name is specified in this property.

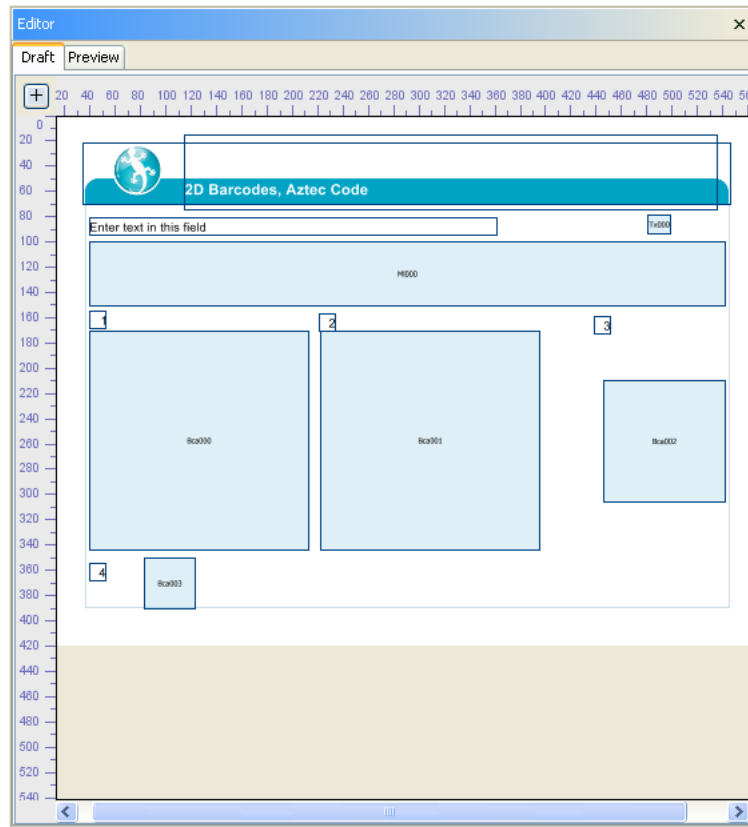
This property is normally deactivated. It gets activated when the **Struct. Append** property is activated. It is, however, optional, and only needed when required by the actual application.

The document *aztec.qdf* (see Fig. 4-115 (Draft mode) and Fig. 4-116 (shown in the Snapform Viewer)) shows various examples of Aztec barcodes.

After selecting the barcode fields in the Snapform Designer, the Properties window displays information about the according properties.

Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig.4-115** Examples of Aztec barcodes from *aztec.qdf*, in Draft mode



**Fig.4-116** Examples of Aztec barcodes from *aztec.qdf*, in the Snapform Viewer



- 1 Bca000: Aztec barcode, Configuration type Full, Stretch content on
- 2 Bca001: Aztec barcode, Configuration type Compact, Stretch content on
- 3 Bca002: Aztec barcode, Configuration type Automatic, Stretch content off, smaller field
- 4 Bca003: Aztec barcode, Configuration type Automatic, Stretch content on, Rune 4

All four barcodes represent the text specified in the entry field, according to the specific properties. In the example file, the text can be changed, and the symbols change accordingly.

#### 4.4.6.3

#### PDF-417 Barcode

The PDF-417 barcode is a high-density “layered” 2-dimensional symbology. The data capacity is 1800 alphanumeric characters or 1100 binary characters per symbol.

The encoded data is available in rows which begin with a Start character, then have a certain number of data characters, and end with a Stop character. This leads to the typical column structure. The number of data character can vary, as well as the number of rows (minimum 3, maximum 90).


The data is represented in code words. A code word consists of four bars and four blanks which are distributed over 17 modules.

With increasing amount of data, the symbol grows in direction of the columns, as more rows are added. When planning a form, be sure that there is sufficient free space for this kind of growth. There is also a necessity for white space around the symbol, so that it can be properly read.

PDF-417 has various error correction levels. With the highest setting (maximum number of error correction code words), only about half of the available data capacity can be used, but a symbol where half of the area is covered or otherwise changed, can still be read.



Another feature of the PDF-417 barcode is the possibility to chain a great number of symbols to represent a large amount of data. This feature is called Macro PDF-417.

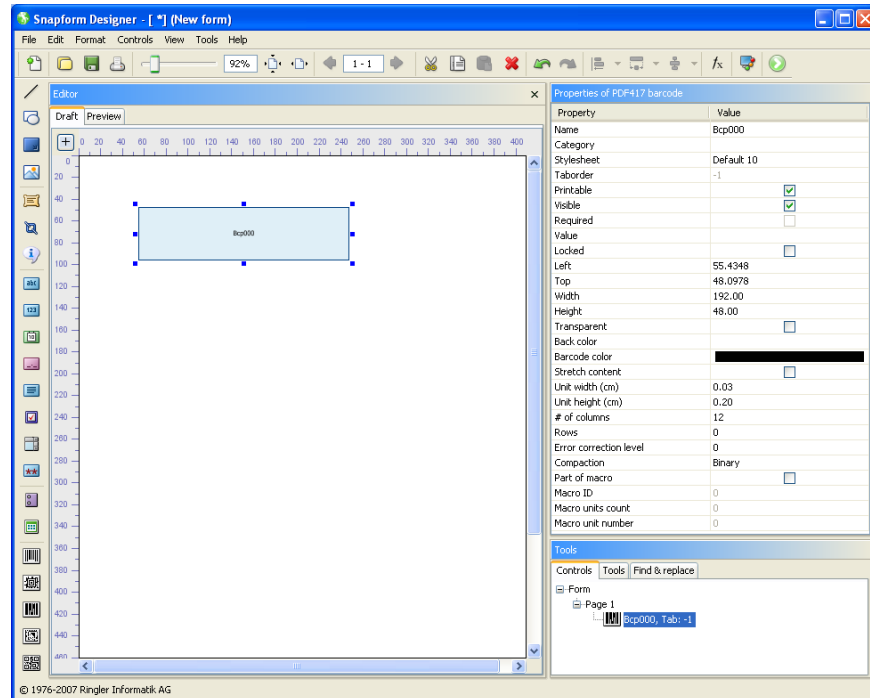
The PDF-417 barcode tool  creates a field which allows displaying a PDF-417 2D barcode. After selecting the tool, a cursor with a 2D barcode field in default size appears (see Fig. 4-117).

**Fig.4-117** PDF-417 barcode tool cursor



Assigned to the PDF-417 barcode tool is a 2D barcode field for PDF-417 code, 192 pt wide and 48 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Bcp** and a three-digit consecutive number, beginning with **000**. The first PDF-417 barcode field placed in a form has therefore the name **Bcp000**. In the object structure it is added to the according page. In Fig. 4-118 the first PDF-417 barcode field of the form has been placed and then selected.


**Fig. 4-118** Newly created  
PDF-417 barcode  
field



The length and the width of the PDF-417 barcode field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the PDF-417 barcode field are controlled in the Properties window (see Fig. 4-119).

**Fig. 4-119** *Properties of a PDF-417 barcode field*

Properties of PDF417 barcode	
Property	Value
Name	Bcp000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	55.4348
Top	48.0978
Width	192.00
Height	48.00
Transparent	<input type="checkbox"/>
Back color	
Barcode color	
Stretch content	<input type="checkbox"/>
Unit width (cm)	0.03
Unit height (cm)	0.20
# of columns	12
Rows	0
Error correction level	0
Compaction	Binary
Part of macro	<input type="checkbox"/>
Macro ID	0
Macro units count	0
Macro unit number	0

## Name

The name of the PDF-417 barcode field. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

### **Stylesheet**

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used.

### **Taborder**

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the barcode is not an active element.

### **Printable**

When the box is checked, the PDF-417 barcode field will be printed.

### **Visible**

When the box is checked, the PDF-417 barcode field is displayed on screen.

### **Required**

This property is always deactivated in PDF-417 barcode fields, and cannot be changed.

### **Value**

In this property of the PDF-417 barcode field, the text to be encoded is entered. Clicking on this property field opens the expression editor. In most of the cases, the value is the result of a calculation.

### **Locked**

When this box is checked, the PDF-417 barcode field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

### **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the PDF-417 barcode field.

<b>Top</b>	Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the PDF-417 barcode field.
<b>Width</b>	Width of the bounding box of the PDF-417 barcode field.
<b>Height</b>	Height of the bounding box of the PDF-417 barcode field.
<b>Transparent</b>	The background of the PDF-417 barcode is set to transparent (no covering background color) with this check box. This property is unchecked by default.
<b>Back color</b>	<p>Background color of the PDF-417 barcode field. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.</p> <p>The background color is only applied when the <b>Transparent</b> check box is unchecked.</p>
<b>Barcode color</b>	<p>Color of the barcode. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.</p> <p><b>Note:</b> <i>Standards for the PDF-417 barcode may contain requirements concerning the colors to be used, and should be referred to. Also note that only the full colors Black, Cyan, Magenta and Yellow are printed as full color (unless specific spot colors are used), and that good printing quality can be achieved only with these colors at 100% saturation, as there will not be screened modules.</i></p>
<b>Stretch content</b>	This check box determines whether the symbol is scaled to the size of the PDF-417 barcode field. When the box is checked, the symbol is scaled to fit into the field perimeter. The value for the Module size property is in this case overridden. When the box is unchecked, the

symbol is displayed using the specified module size, and it is centered in the field.

**Attention:** *When this property is active, the symbol may be scaled down so far with greater amount of data that it cannot be printed out properly anymore, and the resulting module size will be too small.*

**Note:** *When this property is not active, the symbol may grow beyond the field's size and cover other elements of the document.*

### Unit width (cm)

This property controls the width of a module of the PDF-417 barcode. It is entered in cm. As the PDF-417 barcode allows rectangular modules (as opposed to other symbologies which require square modules), this value may differ from the unit height value. It is recommended to select multiples or multiples of one half of the unit height.

The default value is **0.015 cm**. It is highly recommended to not go below **0.075 cm**.

**Note:** *When the **Stretch content** property has been selected, the **Unit width** value is overridden.*

### Unit height (cm)

This property controls the height of a module of the PDF-417 barcode. It is entered in cm. As the PDF-417 barcode allows rectangular modules (as opposed to other symbologies which require square modules), this value may differ from the unit width value. It is recommended to select multiples or multiples of one half of the unit width.

The default value is **0.015 cm**. It is highly recommended to not go below **0.075 cm**.

**Note:** *When the **Stretch content** property has been selected, the **Unit height** value is overridden.*

### # of columns

This property controls how wide the PDF-417 barcode symbol will be. This value sets the number of columns for the payload data. In the actual symbol, 4 additional columns are added.

The smallest value for this property is **3**, and the value should not be higher than **18**, to make sure that the symbol can be properly read.

## Rows

This property specifies the minimum number of rows the symbol must have. If the amount of data is smaller than necessary for the amount of rows, padding characters are added until the required number of rows is reached.

The smallest value for this property is **3**, and the value should not be higher than **90**, to make sure that the symbol can be properly read.

When the value is set to **0**, as many rows are created as are necessary to display the data.

## Rotation angle

The barcode can be rotated by multiples of 90°. Clicking on this property opens a drop-down menu with the selection of the rotation angles **0°**, **90°**, **180°**, **270°**. The default value is 0°.

## Error correction level

This is the level of the error correction. The entered value may be an integer between 0 (no error correction) and 8 (maximum error correction level). This value also determines the amount of payload data which may be encoded in the symbol.

## Compaction

In order to encode the maximum amount of data, it is compressed. The PDF-417 barcode has three compression schemes which are optimized for specific kind of data: numbers, text and binary. The highest compression can be achieved when only numbers are compressed, the lowest degree will be for binary data.

When this property is selected, a drop-down menu pens with the options **Binary**, **Text** and **Numeric**. Default setting is **Text**. This corresponds to the above mentioned compression methods. For a more in-depth description of the compression methods in the PDF-417 barcode, refer to the literature.

When the option **Binary** has been selected, a true binary data stream must be made available for compression. This means that “normal” text must be pre-compressed. In order to do this, the functions **Compress(x)** or **getBytes(string)** can be used. In the example below, the following expression is used for item 4:

**= Compress(M1000)**

**Note:** *When the data does not correspond to the selected compression method, the symbol will not be displayed. When debugging, this should be the first property to be verified.*

## Part of macro

The PDF-417 barcode allows you to chain up to 127 symbols, and achieve an accordingly higher data capacity. This special version of the barcode is called Macro PDF-417. When reading chained symbols, the reading unit stores the data and assembles it. Only when all symbols have been read, will the data be passed on. This appears to subsequent processes, as if a huge symbol had been read.

When this box is checked, the according symbol is part of a chained group. Otherwise, the symbol is self-contained.

When this property has been selected, the properties **Macro unit count** and **Macro unit number** must have a value; they will also be activated in the Properties window. The field for the property **Macro ID** is also activated, but it is not mandatory that it has a value.

## Macro ID

It is possible to specify in advance the file name for the retrieved chained data. That file name is specified in this property.

This property is normally deactivated. It gets activated when the **Part of macro** property is activated. It is, however, optional, and only needed when required by the actual application.

## Macro units count

This property indicates how many symbols there are in the chained group to which the symbol belongs.



Valid values are between **2** and **127**. The value **1** or **0** means the group consists of one single symbol, and therefore, the symbol is self-contained.

This property is normally deactivated. It gets activated when the **Part of macro** property is activated.

## Macro unit number

This property represents the index number of the symbol in the chained group.

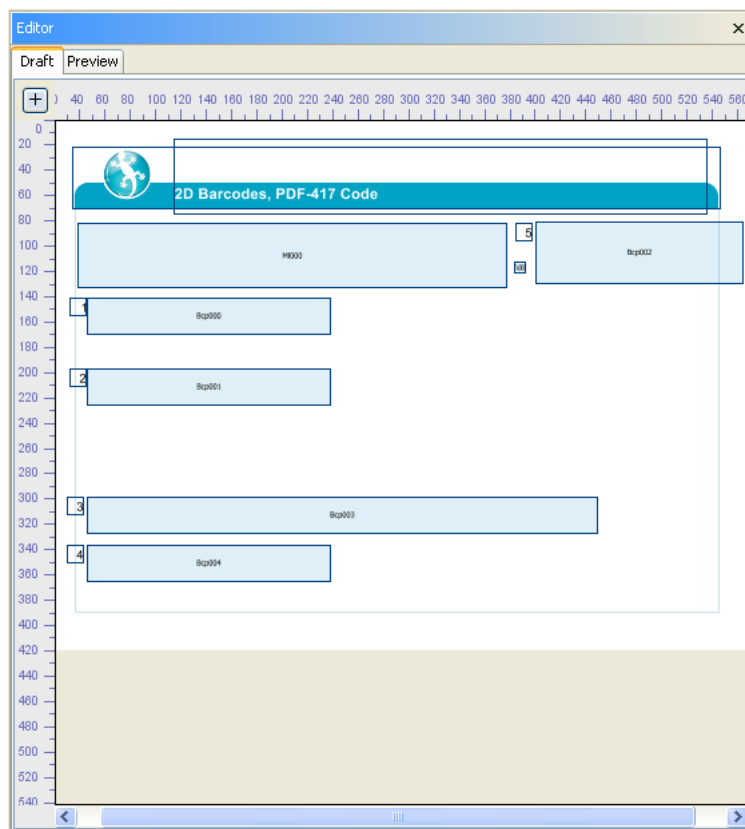
Valid values are between **1** and **127**. The value **0** means that the symbol is not part of a chained group. The value of this property must also be smaller than the value of the **Macro units count** property.

This property is normally deactivated. It gets activated when the **Part of macro** property is activated.

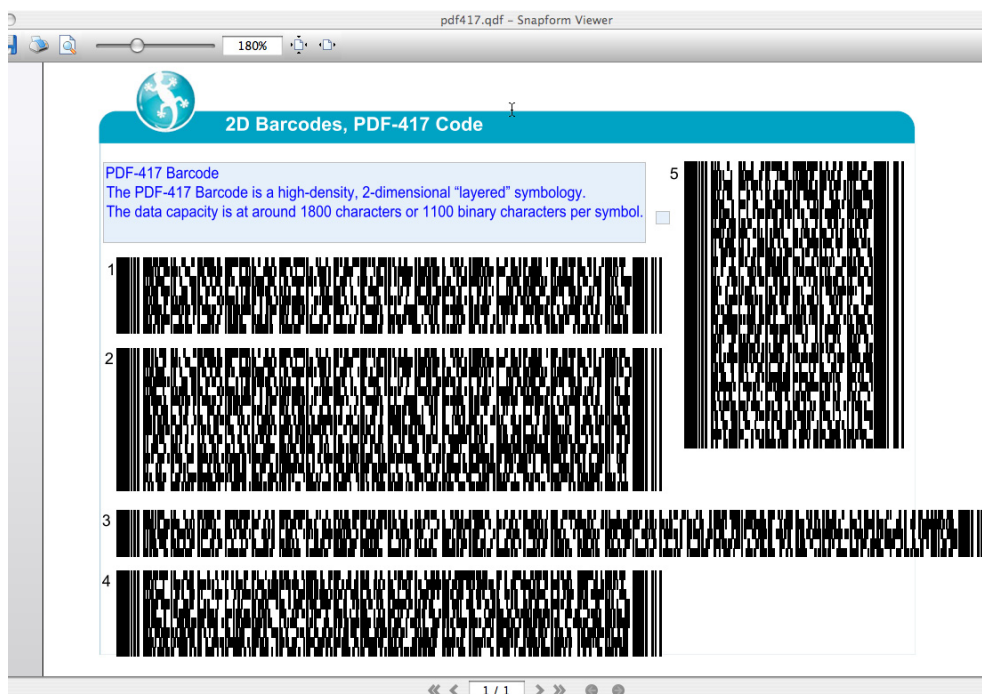
The document *pdf417.qdf* (see Fig. 4-120 (Draft mode) and Fig. 4-121 (shown in the Snapform Viewer)) shows various examples of PDF-417 barcodes.

After selecting the barcode fields in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig. 4-120** Examples of PDF-417 barcodes from pdf417.qdf, in Draft mode



**Fig. 4-121** Examples of PDF-417 barcodes from pdf417.qdf, in the Snapform Viewer



- 1 Bcp000: PDF-417 barcode, 16 columns, error correction level 2
- 2 Bcp001: PDF-417 barcode, 16 columns, error correction level 6
- 3 Bcp002: PDF-417 barcode, 28 columns, error correction level 2
- 4 Bcp004: PDF-417 barcode, 16 columns, error correction level 2, data pre-compressed with **compress(x)**, x
- 5 Bcp003: PDF-417 barcode, 4 columns, error correction level 2

All five barcodes represent the text specified in the entry field, according to the specific properties. In the example file, the text can be changed, and the symbols change accordingly.

#### 4.4.6.4

#### Datamatrix


The Datamatrix barcode is high-density 2D barcode symbology with square modules. The maximum data capacity is 3116 numeric characters, 2335 alphanumeric characters or 1556 Bytes of binary data. The encoded data consists of square zones, and the symbol itself is normally also square-shaped. Certain rectangular forms are possible, however.

The square zones are framed by a so-called finding pattern, which is also used for recognizing the orientation of the symbol. The symbol requires blank space around it, in order to be read properly.

The size and configuration (including error correction information) of the square zones is specified in the standards. Datamatrix symbols cannot grow dynamically; when the amount of data exceeds the specific value for the symbol size, the next larger symbol must be selected. In Snapform, the symbol selection can be done automatically.

The Datamatrix barcode can encode 256 characters of the “normal” ASCII character set. Using so-called Extended Channels, other character sets can be encoded. Switching between character sets within the symbol is possible with special control characters. Such a character set switch must however be supported by the application.

According to the standard for the Datamatrix barcode, it is possible to structurally chain up to 16 symbols. This option is, however, not implemented in Snapform.

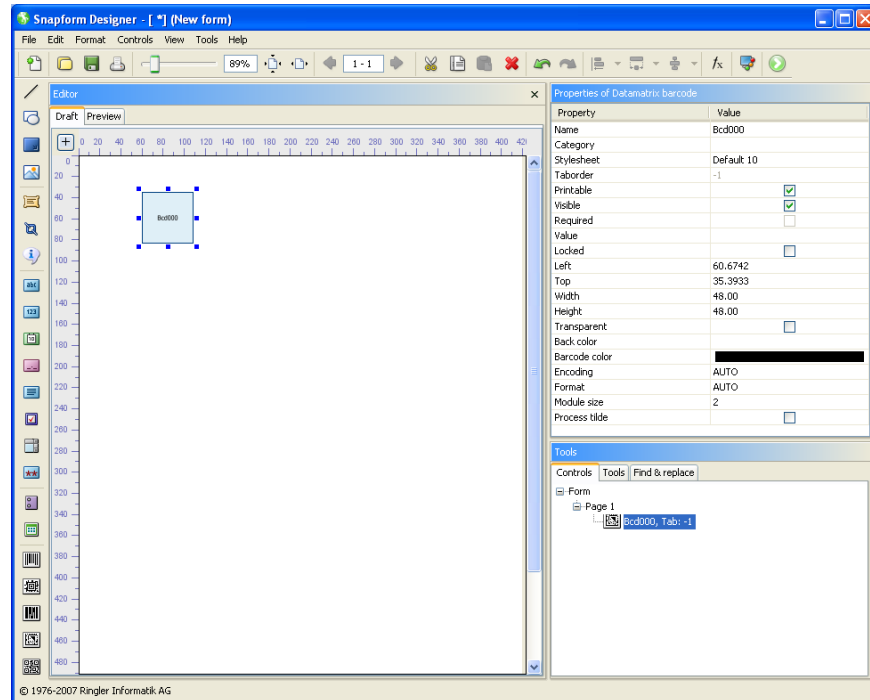
The Datamatrix barcode tool  creates a field which allows displaying a Datamatrix 2D barcode. After selecting the tool, a cursor with a 2D barcode field in default size appears (see Fig. 4-122).

**Fig. 4-122** *Datamatrix  
barcode tool cursor*



Assigned to the Datamatrix barcode tool is a 2D barcode field for Datamatrix code, 48 pt wide and 48 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Bcd** and a three-digit consecutive number, beginning with **000**. The first Datamatrix barcode field placed in a form has therefore the name **Bcd000**. In the object structure it is added to the according page. In Fig. 4-123 the first Datamatrix barcode field of the form has been placed and then selected.

**Fig. 4-123** Newly created  
Datamatrix  
barcode field




The length and the width of the Datamatrix barcode field can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

**Note:** *The field will automatically be scaled to a square when an anchor point is moved. The longer side determines the size of the square.*

The other features of the Datamatrix barcode field are controlled in the Properties window (see Fig. 4-124).

**Fig. 4-124** *Properties of a Datamatrix barcode field*

Properties of Datamatrix barcode	
Property	Value
Name	Bcd000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	60.6742
Top	35.3933
Width	48.00
Height	48.00
Transparent	<input type="checkbox"/>
Back color	
Barcode color	
Encoding	AUTO
Format	AUTO
Module size	2
Process tilde	<input type="checkbox"/>

## Name

The name of the Datamatrix barcode field in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used.

## **Taborder**

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the barcode is not an active element.

## **Printable**

When the box is checked, the Datamatrix barcode field will be printed.

## **Visible**

When the box is checked, the Datamatrix barcode field is displayed on screen.

## **Required**

This property is always deactivated in Datamatrix barcode fields, and cannot be changed.

## **Value**

In this property of the Datamatrix barcode field, the text to be encoded is entered. Clicking on this property field opens the expression editor. In most of the cases, the value is the result of a calculation.

## **Locked**

When this box is checked, the Datamatrix barcode field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Datamatrix barcode field.

## **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the Datamatrix barcode field.

## **Width**

Width of the bounding box of the Datamatrix barcode field.

If its value is smaller, the height is automatically adjusted to the width, in order to create a square field. Otherwise, the Width value is not changed.

## Height

Height of the bounding box of the Datamatrix barcode field.

If its value is smaller, the width is automatically adjusted to the height, in order to create a square field. Otherwise, the Height value is not changed.

## Transparent

The background of the Datamatrix barcode is set to transparent (no covering background color) with this check box. This property is unchecked by default.

## Back color

Background color of the Datamatrix barcode field. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows you to specify the color.

The background color is only applied when the **Transparent** check box is unchecked.

## Barcode color

Color of the barcode. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

**Note:** *Standards for the Datamatrix barcode may contain requirements concerning the colors to be used, and should be referred to. Also note that only the full colors Black, Cyan, Magenta and Yellow are printed as full color (unless specific spot colors are used), and that good printing quality can be achieved only with these colors at 100% saturation, as there will not be screened modules.*

## Rotation angle

The barcode can be rotated by multiples of 90°. Clicking on this property opens a drop-down menu with the selection of the rotation angles **0°**, **90°**, **180°**, **270°**. The default value is 0°.

## Encoding

This property specifies the method which is used to encode the data stream for the symbol. When selecting this property, a drop-down menu opens with the options **AUTO**, **ASCII**, **C40**, **TEXT** and **BASE256**. **AUTO** the default. The settings have the following meaning:



## **AUTO**

The best possible encoding is selected automatically by the software.

## **ASCII**

This encoding is used when the data stream consists mainly of characters with ASCII values between 0 and 127. It encodes one alphanumeric or two numeric characters into one Byte.

## **C40**

This encoding is used when the data stream consists mainly of numbers and uppercase letters. It encodes three alphanumeric characters into two Bytes.

## **TEXT**

This encoding is used when the data stream consists mainly of numbers and lowercase letters. It encodes three alphanumeric characters into two Bytes.

## **BASE256**

This encoding is used when the data stream consists of 8-bit binary data.

Normally, the setting AUTO creates sufficiently well encoded symbols.

## **Format**

The Datamatrix barcode has a discrete series of dimensions, measured in the number of modules per side. These dimensions are defined by the internal structure of the symbol. The following dimensions are defined and allowed:

1. square

10x10, 12x12, 14x14, 16x16, 18x18, 20x20, 22x22, 24x24, 26x26, 32x32, 36x36, 40x40, 44x44, 48x48, 52x52, 64x64, 72x72, 80x80, 88x88, 96x96, 104x104, 120x120, 132x132, 144x144

2. rectangular

8x18, 8x32, 12x26, 12x36, 16x36, 16x48

When this property is selected, a drop-down menu opens with the options **AUTO**, followed by the list of the above mentioned dimensions. **AUTO** is the default.

The option **AUTO** means that based on the encoded data, the best possible symbol size is selected. The symbol is anchored to the upper left corner of the barcode field, and therefore, its bottom right corner moves when the symbol grows.

**Note:** *Even when a fixed dimension is selected, the next larger size is used when the amount of data exceeds the maximum for the size. It is therefore important to estimate the amount of data to be encoded, and to reserve sufficient white space around the symbol.*

**Note:** *The rectangular symbols have (as it shows in the list) a small data capacity. The biggest rectangular symbol (16x48) can at most encode 72 alphanumeric characters or 47 Byte worth of binary data. When this amount of data is exceeded, a suitable square format is used.*

## Module size

This is the size of a module (the smallest unit of the Datamatrix barcode). Measuring unit is Point, and the default value is 2. This value should not be much smaller, as otherwise, there may be problems in printouts.

## Process tilde

In order to represent control characters which cannot be entered via keyboard, the Tilde character (~) is used. This property specifies whether the Tilde character is regarded as a control character.

When this box is checked, the Tilde character is interpreted as a control character. This has the following implications which must be taken into consideration when the encoding data stream is calculated:

- ~~ is replaced with the actual tilde character (therefore double the Tilde character if one must get into the data stream).
- ~dxxx Is replaced with the character whose ASCII code is xxx (which means that, for example, ~d065 will be replaced with **A**). This representation is important when control characters in the lowest ASCII range are encoded.
- ~1 will be replaced with the **FNC1** character. This is a special character which can be at the very beginning of a Datamatrix

barcode datastream, and has the meaning that the data is conforming to the UCC/EAN Standard Identifier Format. This character can only be placed at the first position of the datastream.

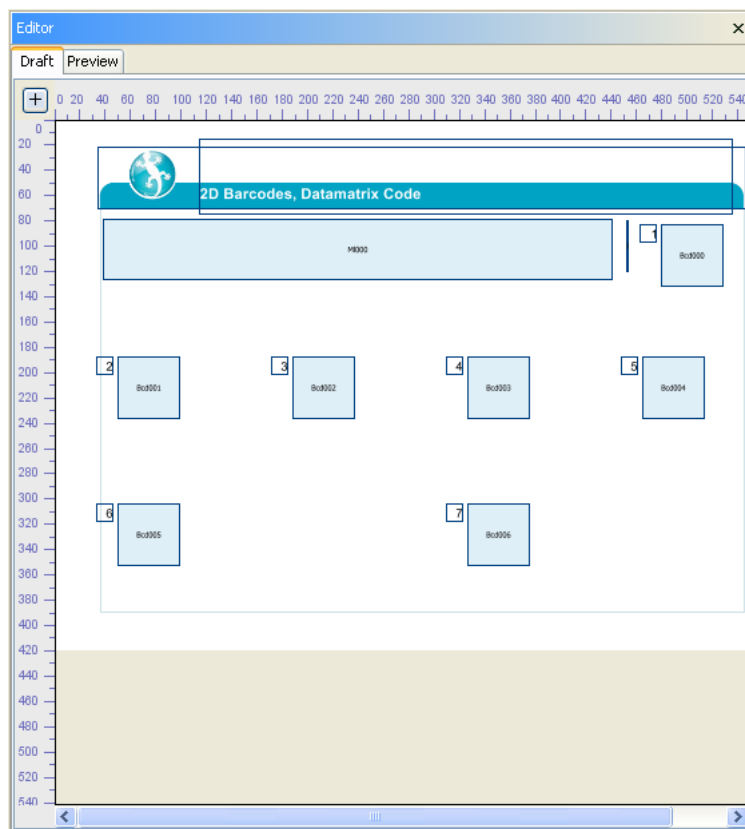
- **~3** is a special character which can be at the very beginning of a Datamatrix barcode datastream, and has the meaning that the data contains control functions for the reading unit.
- **~Exxxxxx** is replaced with the so-called Extended Interpretation Channel flag with number **xxxxxx**, a control character which can be evaluated on application level, and that it is used often for the indication of switching character sets.

This property is deactivated in the default setting.

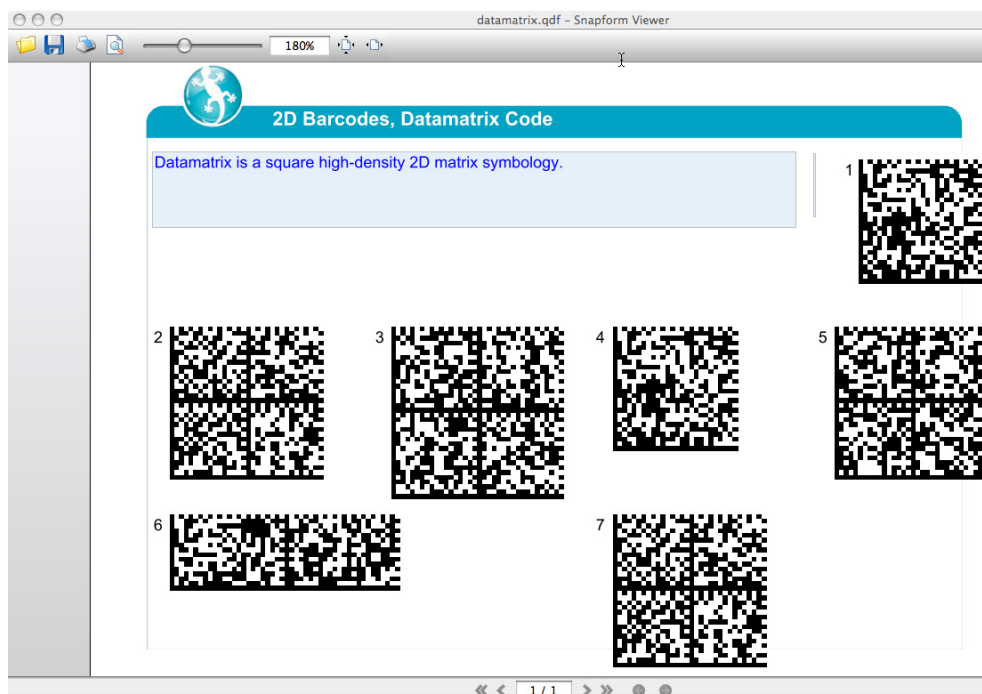
The document *datamatrix.qdf* (see Fig. 4-125 (Draft mode) and Fig. 4-126 (shown in the Snapform Viewer)) shows various examples of Datamatrix barcodes.

After selecting the barcode fields in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig. 4-125** Examples of Datamatrix barcodes from *datamatrix.qdf*, in Draft mode



**Fig. 4-126** Examples of Datamatrix barcodes from *pdf417.qdf*, in the Snapform Viewer



- 1 Bcd000: Datamatrix barcode, Format and Encoding automatic
- 2 Bcd001: Datamatrix barcode, Format automatic, Encoding ASCII
- 3 Bcd002: Datamatrix barcode, Format automatic, Encoding C40
- 4 Bcd003: Datamatrix barcode, Format automatic, Encoding TEXT
- 5 Bcd004: Datamatrix barcode, Format automatic, Encoding BASE256
- 6 Bcd005: Datamatrix barcode, Format 16x48, Encoding TEXT
- 7 Bcd006: Datamatrix barcode, Format 16x48, Encoding ASCII

All seven barcodes represent the text specified in the entry field, according to the specific properties. In the example file, the text can be changed, and the symbols change accordingly.

Item 7 should display a rectangular symbol, but the selected encoding creates more characters than allowed. Therefore, a square symbol is created. When the entered text is changed to **Datamatrix is a 2-D Matrix symbology** the rectangular symbol appears.


#### 4.4.6.5

#### QRCode

The QRCode barcode is a high-density square matrix symbology. The data capacity is 7089 numeric characters, 4296 alphanumeric characters, 1817 Kanji characters or 2953 Bytes worth of binary data (maximum symbol size, lowest error correction level). The QRCode has been designed for applications in Japan. The symbol contains a typical “finder pattern” (a larger symbol in the top left, top right and bottom left corner, plus a smaller one near the bottom right corner), which allows omnidirectional reading.

The symbol can contain one of 40 pre-defined sizes, and several error correction levels are provided.

Besides characters with a decimal code between 0 and 255, Kanji characters are also supported (Hex values 8140 - 9FFC and E040 - EBBF).

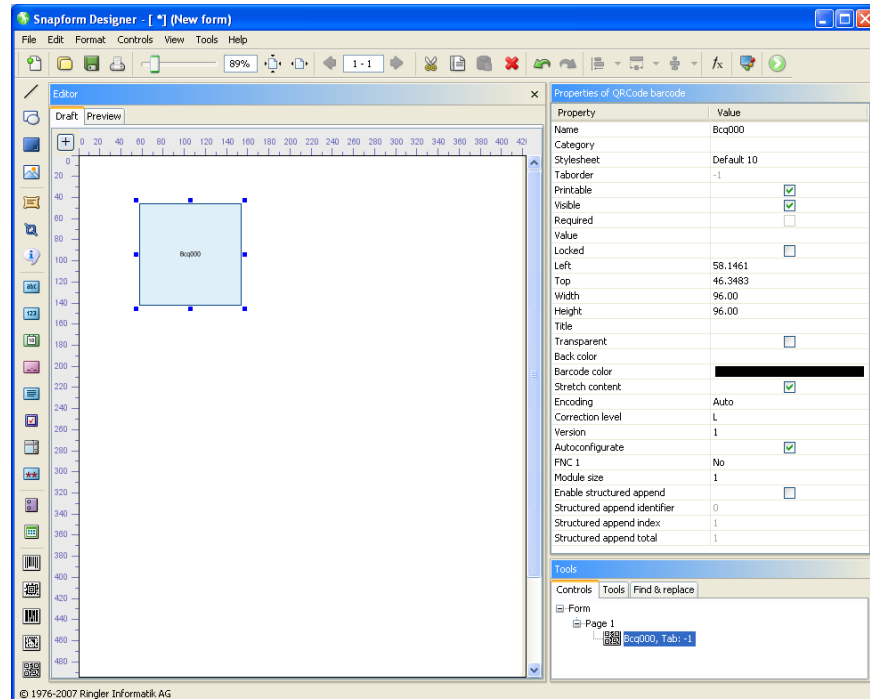
The QRCode barcode tool  creates a field which allows displaying a QRCode 2D barcode. After selecting the tool, a cursor with a 2D barcode field in default size appears (see Fig. 4-127).

**Fig. 4-127** *QRCode barcode tool cursor*



Assigned to the QRCode barcode tool is a 2D barcode field for QRCode code, 96 pt wide and 96 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Bcq** and a three-digit consecutive number, beginning with **000**. The first QRC barcode field placed in a form has therefore the name **Bcq000**. In the object structure it is added to the according page. In Fig. 4-128 the first QRCode barcode field of the form has been placed and then selected.

**Fig. 4-128** *Newly created QRCode barcode field*




The length and the width of the QRCode barcode field can be changed by dragging the anchor points of the bounding box. The position of the

rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the QRCode barcode field are controlled in the Properties window (see Fig. 4-129).

**Fig. 4-129** *Properties of a  
QRCode barcode  
field*

Properties of QRCode barcode	
Property	Value
Name	Bcq000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	58.1461
Top	46.3483
Width	96.00
Height	96.00
Title	
Transparent	<input type="checkbox"/>
Back color	
Barcode color	
Stretch content	<input checked="" type="checkbox"/>
Encoding	Auto
Correction level	L
Version	1
Autoconfigure	<input checked="" type="checkbox"/>
FNC 1	No
Module size	1
Enable structured append	<input type="checkbox"/>
Structured append identifier	0
Structured append index	1
Structured append total	1

## Name

The name of the QRCode barcode field in the document. This name can be changed.



## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used.

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence, which is logical, because the barcode is not an active element.

## Printable

When the box is checked, the QRCode barcode field will be printed.

## Visible

When the box is checked, the QRCode barcode field is displayed on screen.

## Required

This property is always deactivated in QRCode barcode fields, and cannot be changed.

## Value

In this property of the QRCode barcode field, the text to be encoded is entered. Clicking on this property field opens the expression editor. In most of the cases, the value is the result of a calculation.

## Locked

When this box is checked, the QRCode barcode field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

<b>Left</b>	X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the QRCode barcode field.
<b>Top</b>	Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the QRCode barcode field.
<b>Width</b>	Width of the bounding box of the QRCode barcode field.
<b>Height</b>	Height of the bounding box of the QRCode barcode field.
<b>Title</b>	<p>This is a line of text which is placed above the QRCode barcode field. This title may, for example, contain information for the operator who manually scans in the code.</p> <p>The title may be either simple text, or entered in HTML form. It is of fixed font size (10 pt) and its length is limited to the width of the QRCode barcode field.</p>
<b>Transparent</b>	The background of the QRCode barcode is set to transparent (no covering background color) with this check box. This property is unchecked by default.
<b>Back color</b>	<p>Background color of the QRCode barcode field. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.</p> <p>The background color is only applied when the <b>Transparent</b> check box is unchecked.</p>
<b>Barcode color</b>	Color of the barcode. By clicking on this property, the color picker opens (see section 3.2.9.2), which allows to specify the color.

**Note:** *Standards for the QRCode barcode may contain requirements concerning the colors to be used, and should be referred to. Also note that only the full colors Black, Cyan, Magenta and Yellow are printed as full color (unless specific spot colors are used), and that good printing quality can be achieved only with these colors at 100% saturation, as there will not be screened modules.*

## Stretch content

This check box determines whether the symbol is scaled to the size of the QRCode barcode field. When the box is checked, the symbol is scaled to fit into the field perimeter. The value for the Module size property is in this case overridden. When the box is unchecked, the symbol is displayed using the specified module size, and it is aligned to the top left corner the field.

**Attention:** *When this property is active, the symbol may be scaled down so far with greater amount of data that it cannot be printed out properly anymore, and the resulting module size will be too small.*

**Note:** *When this property is not active, the symbol may grow beyond the field's size and cover other elements of the document.*

## Encoding

This property specifies the method which is used to encode the data stream for the symbol. When selecting this property, a drop-down menu opens with the options **Auto**, **Byte**, **Numeric**, **Alphanumeric** and **Kanji**. **Auto** is the default. With the **Auto** setting the most suitable encoding method for the given data stream will be used. The settings have the following meaning:

### Byte

The encoding occurs bitwise. In order to enable this encoding method, the data stream must be available as a byte sequence. This can be done using the functions `getBytes(string)` and `Compress(x)`.

### Numeric

The encoding occurs only for the numbers **0 - 9**.

## Alphanumeric

The encoding occurs only for the numbers **0 - 9**, upper case letters **A - Z** and 9 special characters **<Space> \$ % \* + - . / :**

## Kanji

The encoding occurs only for Japanese characters (Kanji, including Hiragana and Katakana) with Hex code **8140 - 9FFC** and **E040 - EBBF**. In this range, numbers, upper case letters and certain special characters are also contained.

When the data stream contains unencodeable characters, the QRCode barcode field will be displayed as a grey area.

## Correction level

The QRCode barcode has several error correction levels. The lowest level allows the loss of about 7% of the symbol, the highest level about 30%.

When this property is selected, a drop-down menu opens with the options **L, M, Q** and **H**. Default value is **L**.

## Version

In the QRCode barcode terminology, the term "Version" stands for the size of the symbol. 40 sizes are defined which may be available for the encoding.

When this property is selected, a drop-down menu opens with the options corresponding to the 40 different sizes (values between **1** and **40**). Default value is **1**.

When the property **Autoconfigure** is selected, the next larger symbol is selected with increasing amount of data. When it is not selected, the QRCode barcode field is displayed as a grey area when the amount of data is too big.

## Autoconfigure

When this box is checked, the QRCode barcode field will select automatically the next larger symbol with increasing amount of data. The symbol grows accordingly (or the module size shrinks, if the **Stretch content** option is active). Default for this property is unchecked.

## FNC 1

The QRCode barcode has a special control character FNC1, which can be used for particular purposes. It is either at the first or the second position of the data stream.

When this property is selected, a drop-down menu opens with the options **No**, **1** and **2**. Default value is **No**.

The option **No** means that no FNC1 character is used. The option **1** means that the FNC1 character is at the first position in the data stream, which means that the encoded data is formatted according to the UCC/EAN Application Identifiers standards. The option **2** means that the FNC1 character is at the second position in the data stream, which means that the data is formatted according to separately specified standards which are registered at AIM International.

## Module size

This is the size of a module (the smallest unit of the QRCode barcode). Measuring unit is Point, and the default value is 1. This value should not be much smaller, as otherwise, there may be problems in printouts.

**Note:** *When the **Stretch content** property has been selected, the **Module size** value is overridden.*

## Enable structured append

The QRCode barcode allows chaining up to 16 symbols, and achieve an accordingly higher data capacity. When reading chained symbols, the reading unit stores the data and assembles it. Only when all symbols have been read is the data passed on. This appears to subsequent processes, as if a huge symbol had been read.

When this box is checked, the according symbol is part of a structurally chained group. Otherwise, the symbol is self-contained.

When this property has been selected, the properties **Structured append total** and **Structured append index** must have a value; they will also be activated in the Properties window. The field for the Structured append identifier is also activated, but it is not mandatory for it to have a value.

## Structured append total

This property indicates how many symbols there are in the structurally chained group to which the symbol belongs.

Valid values are between **2** and **16**. The value **1** means that the group consists of one single symbol, and therefore, the symbol is self-contained.

This property is normally deactivated. It gets activated when the **Enable structured append** property is activated.

## Structured append index

This property represents the index number of the symbol in the structurally chained group.

Valid values are between **1** and **16**.

This property is normally deactivated. It gets activated when the **Enable structured append** property is activated.

## Structured append identifier

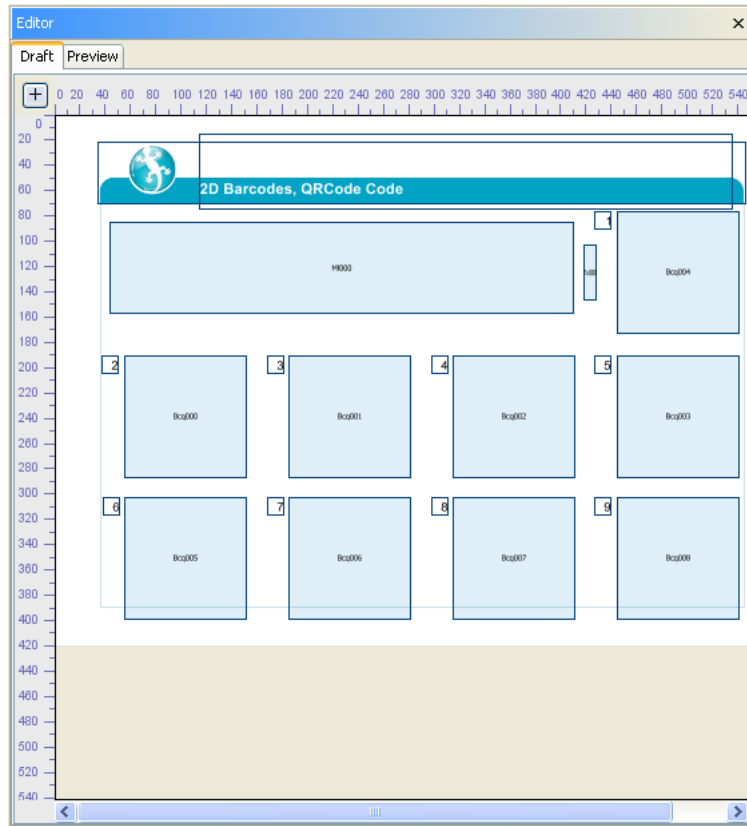
It is possible to specify in advance the file name for the retrieved chained data. That file name is specified in this property.

This property is normally deactivated. It gets activated when the **Enable structured append** property is activated. It is, however, optional, and only needed when required by the actual application.

The document *qrcode.qdf* (see Fig. 4-115 (Draft mode) and Fig. 4-116 (shown in the Snapform Viewer)) shows various examples of QRCode barcodes.

After selecting the barcode fields in the Snapform Designer, the Properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the expression (if there is any).

**Fig.4-130** Examples of  
QRCode barcodes  
from *qrcode.qdf*, in  
Draft mode



**Fig.4-131** Examples of  
QRCode barcodes  
from *qrcode.qdf*, in  
the Snapform  
Viewer



- 1 Bcq004: QRCode barcode, Format and Encoding Automatic
- 2 Bcq000: QRCode barcode, encoding Binary; for calculating the data stream, the function **getBytes(MI000)** was used.
- 3 Bcq001: QRCode barcode, Encoding Numeric
- 4 Bcq002: QRCode barcode, Encoding Alphanumeric
- 5 Bcq003: QRCode barcode, Encoding Kanji
- 6 Bcq005: QRCode barcode, Encoding Automatic, Error correction level L
- 7 Bcq006: QRCode barcode, Encoding Automatic, Error correction level M
- 8 Bcq007: QRCode barcode, Encoding Automatic, Error correction level Q
- 9 Bcq008: QRCode barcode, Encoding Automatic, Error correction level H

All barcodes represent the text specified in the entry field, according to the specific properties. In the example file, the text can be changed, and the symbols change accordingly.

The item 5 symbol (encoding Kanji) requires entries in that particular character set range.

## 4.4.7

### Tables

In many forms, data is represented in tabular form. Often the number of rows of the table is variable and cannot be predicted. With paper forms, an optimum number of rows must be assumed (which more often depends on the available space). When the table needs more rows, attachments must be used.

In electronic forms, there is the possibility to allow tables to grow, and to “move down” the subsequent form elements. This approach is viable for on-screen versions of the form, because scrolling is necessary anyway. The problem appears when the form must be printed out nevertheless. The pagination is often an integral part of the form, and may not be modified easily.

Snapform has a great emphasis on the correct representation of a page. Therefore, when a data overflow happens with a table, an additional

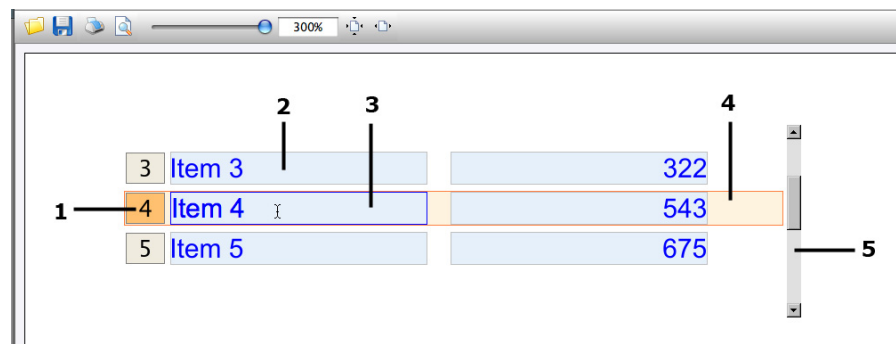


page is created, and in the actual form, summarized data of the table can be shown.

The base for tables in Snapform is a Table container to which the individual table elements are assigned. The behavior of the table is determined in the properties of the Table container.

Tables have various visual elements in Snapform, which are shown in Fig. 4-132.

**Fig. 4-132** *Elements of a table in Snapform*




- 1 Selector: The purpose of this element is selecting a whole row of the table. The selector is numbered, where the initial value is a property of the table.
- 2 Field in the table: Within the table it is possible to navigate using the cursor keys.
- 3 Active field: This field is active (meaning that it has the focus).
- 4 Active row: This row has become active when the field (item 3) has been activated.
- 5 Scrollbars: When there are more table rows than can be displayed in the available space, a scrollbar appears, allowing to navigate within the table.

#### 4.4.7.1

### The Table tool

The Table container is a field which makes every field placed within its perimeter a part of the table group. In the object structure, these fields are one level below the other fields. The table container is invisible outside of the Draft view (in Preview and in the Snapform Viewer).

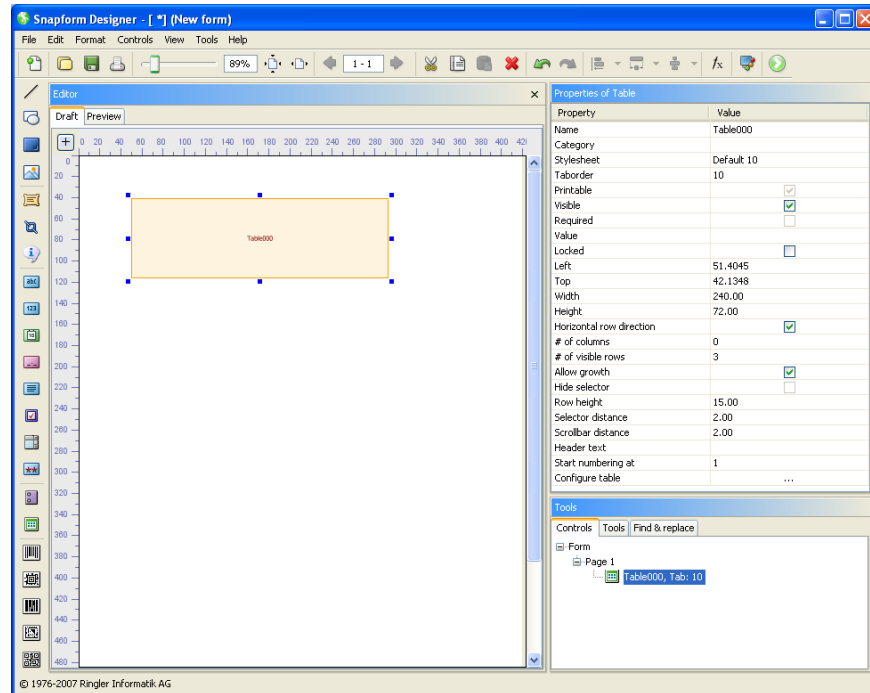
The Table tool  creates a table container. After selecting the tool a cursor with a table container in default size appears (see Fig. 4-133).

**Fig.4-133** *Table container cursor*



Assigned to the Table tool is a table container, 240 pt wide and 72 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **Table** and a three-digit consecutive number, beginning with **000**. The first table container placed in a form has therefore the name **Table000**. In the object structure it is added to the according page. In Fig. 4-134 the first table container of the form has been placed and then selected.

**Fig. 4-134** *Newly created table container*



The length and the width of the table container can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by “grabbing” and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When “grabbing” and pressing the mouse button, the cursor changes to an outlined arrow.

The next step to create a table is adding table elements (see section 4.4.7.2).

The other features of the table container are controlled in the Properties window (see Fig. 4-135).

**Fig. 4-135** *Properties of a table container*

Properties of Table	
Property	Value
Name	Table000
Category	
Stylesheet	Default 10
Taborder	10
Printable	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	51.4045
Top	42.1348
Width	240.00
Height	72.00
Horizontal row direction	<input checked="" type="checkbox"/>
# of columns	0
# of visible rows	3
Allow growth	<input checked="" type="checkbox"/>
Hide selector	<input type="checkbox"/>
Row height	15.00
Selector distance	2.00
Scrollbar distance	2.00
Header text	
Start numbering at	1
Configure table	...

## Name

The name of the table container in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For tables the stylesheet settings in the **Tables** tab are of great importance.

## Taborder

Sequence number of the element in the tab order. This value is assigned automatically, and is in the order of the field tab sequence (beginning with 10 and incremented in steps of 10). The first field of the form has tab order value 10, the second the value 20, etc. The tab order value can be changed, so that a specific tabbing sequence can be created which has no relationship with the order the fields have been added.

With tables, make sure that the table container has the lowest taborder number of the whole table group.

## Printable

This check box is unchecked and cannot be changed. Table containers are never printed.

## Visible

When this box is checked, the table container including all assigned field elements are displayed on screen.

## Required

This property is always deactivated, and cannot be changed.

## Value

This property is shown with table containers, but not active, and it cannot be activated.

## Locked

When this box is checked, the table container and all its contained form elements get locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive (also for the check boxes within the container). Access to these properties is only available after deactivating this check box.

The form elements assigned to the table container can be selected, and their properties can be displayed, but it is not possible to change them.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the table container.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the table container.

## Width

Width of the bounding box of the table container.

## Height

Height of the bounding box of the table container.

## Horizontal row direction

This property defines the direction in which a “logical row” runs. A “logical row” corresponds to the logical group of fields in the table. Normally, the logical row is horizontal, but it may also be vertical (such as in a classical timetable field).

**Note:** A “Visual row” is, however, as the name states, purely visual, and it may not have a context with the information displayed in the table. A visual row is always horizontal, and a visual column is always vertical. In the following explanations, it is stated whether a logical or a visual row or column is referred to.

When this box is checked, the logical row runs horizontally, and additional rows are added below the last current row. In the other case, the logical row runs vertically, and additional “rows” are added to the right of the last current logical row. This box is checked as default.

**Note:** When this property is switched, be aware that certain other properties are “rotated”. In particular, be aware that the **Row height** property value now represents the visual column width, and that it must therefore be accordingly adjusted.

## # of columns

This property specifies how many columns are defined in the table. This value corresponds essentially to the number of form elements which are

placed in the container (check boxes which are part of a radio button group count individually). When additional elements are added to the table container, this value changes automatically.

This value is for information purposes only. It may be changed, but the change is discarded immediately.

## # of visible rows

This property specifies how many logical rows of the table are displayed at a time. Default value is 3.

The actual displayed number of rows depends on the value of the **Row height** property and the dimensions of the table container. When the table container is not big enough, only as many rows are displayed as will fit within the perimeter of the table container. Rows may be displayed cropped.

A special case is when this value is set to **1**. This leads to a so-called “phantom table”. As only one row is available, entries may not be made correctly, and also the display of the information is limited. In order to display data, an auxiliary table, named “report” is used which gives access to all data in the table. This table is defined in the **Configure table** property.

## Allow growth

Snapform allows you to build up tables with no pre-determined number of logical rows. This property controls whether rows may be added beyond the number specified in the **# of visible rows** property. When the actual number of logical rows goes beyond the number of visible rows, a scrollbar appears which allows scrolling up and down the displayed rows.

When this box is checked, an unlimited number of rows can be added to the table. Otherwise, the number of rows of the table is limited.

When more rows are available than can be displayed, the Snapform Viewer creates additional pages when printing (or in the Print preview), which contain all table values in a simple tabular form. This table is called

“report”. The layout of this table is specified in the **Configure table** property.

## Row height

This property specifies the height of the logical row of the table. This value sets the distance in direction of the column between two corresponding fields in logical rows. Normally, this is the distance between the visual rows. Measuring unit is Points and the default value is **15** (which matches well the default value of the height of entry fields).

**Helpful Hint:** *In order to prevent overlaps when the table is displayed, this value may need to be adjusted. As a rule of thumb, “Height of the bounding box around all fields of a logical row plus 3 Points” is a useful value. When all fields are properly aligned on a horizontal line, this also means “Height of the tallest field in the row plus 3 Point”*

**Note:** *When the property **Horizontal row direction** is unchecked, this value must be verified. In order to prevent overlapping, the above rule of thumb has to be applied to the field width.*

## Selector distance

The selector (see Fig. 4-132) is used to mark and select logical rows. It is displayed automatically at the left (or above) the logical row, outside of the perimeter of the table container.

This property controls the distance between the selector and the border of the table container. Measuring unit is Points and the default value is 2 pt. The selector has the same height as the field of the first table column and is approximately square, so that two digits can be displayed without scaling.

These dimensions must be taken into account when placing tables.

## Scrollbar distance

When the **Allow growth** property is active, and the table contains more rows than can be displayed, a scrollbar appears at the right side of (or below) the table, which allows navigating between the logical rows (see Fig. 4-132).



This property controls the position of the scrollbar in relationship to the table container. Measuring unit is Points and the default value is 2 Pt. The appearance of the scrollbar is provided by the operating system and cannot be controlled from Snapform.

These dimensions must be taken into account when placing tables. It is recommended that you test the form on all platforms on which it can be expected to be used.

### **Header text**

This property specifies the header which gets printed on the listing tables. This value appears also in the header of the data entry table of phantom tables.

Clicking on the property opens the expression editor. The value can either be a fixed text, or the result of a calculation. This property supports only simple text; HTML tags are not interpreted.

### **Start numbering at**

This property specifies the initial value of the line numbering in the selectors (see Fig. 4-132). Default value is **1**. Only integers can be entered.

### **Configure table**

This property specifies the table called "report" (or "listing"). This table is displayed in print when the table in the form contains more logical rows than can be displayed. It is also used as data entry table for phantom tables.

Clicking on ... for this property opens the configuration window (see Fig. 4-136).

**Fig. 4-136** Configuration window for report table

Column	Tab...	Width	Header	S...
Nr004	190	33	Number	<input checked="" type="checkbox"/>
Cb002	240	12		<input type="checkbox"/>
Tx005	200	122	Item	<input checked="" type="checkbox"/>
Lb019	-1	60		<input type="checkbox"/>
Lb018	-1	14		<input type="checkbox"/>
Nr005	210	61	Unit price	<input type="checkbox"/>

**Options**

Column for message: 2

Style for report title: Arial 10

Style for report headers: Arial 10

Print report after page: 1

Orientation: ☒ Portrait ☐ Landscape

Buttons: Help, Close

The configuration window consists of two zones, the table definition and the report properties. The table definition zone is a five-column table to specify the listing table.

## Column

Name of the form element of the according column.

**Attention:** When the name of the column is changed in this table, the name of the form element gets changed too. This might lead to errors in expression.

## Taborder

Order number of the form element of the according column.

**Attention:** The order number not only controls the tab order for the data entry table, but also the one in the form itself. When this value is changed, the whole tab order of the form is modified. Chances are that the same order number may be used twice.

## Width

Width of the according column in the report table. Base value is width of the according form element. When displaying in the listing window or in the print view, the columns are scaled proportionally to the total width. It is possible to enter the percentage of the total width into this column.

## Header

This is the title above the column. It appears in the data entry table as well as in the print view. This text can be freely chosen, and it has no equivalent on the form itself.

## Show total

This check box has a double function. When it is checked, and there is a phantom table, a button to open the data entry window appears to the left of the according field. In general (at least with columns from Number fields), the total sum of the entered values is calculated and displayed in the form, in the data entry window, and in the print view.

**Note:** *In order to display a button to open the data entry window in phantom tables, at least one of these check boxes must be checked.*

Below this table, various properties of the report table are shown.

## Column for message

In the print view of a form with tables containing more rows than can be displayed, the "report" is created. In this case, no detail data is shown in the table in the form, but a message "see listing" is shown. This text appears also in phantom tables when there is more than one row of data.

This property specifies the column in which this text is shown. Clicking on this field opens a drop-down menu with the numbers of the table columns. The message text appears in the selected column. The message text is hard coded and cannot be changed.

**Note:** *For phantom tables keep the following points in mind: 1. The form element for the according column must be able to display text. 2. In order to make the text appear, the check box **Show total** must have been checked for that particular column.*

## Style for report title

This property specifies the stylesheet used for the title of the report table. Primarily the font information defined in the stylesheet is applied. Clicking on this property opens a drop-down menu containing the list of the stylesheets available for the form.

The text of the header is specified in the **Header text** property of the table container.

### Style for report headers

This property specifies the stylesheet used for the column headers of the report table. Primarily the font information defined in the stylesheet is applied. Clicking on this property opens a drop-down menu containing the list of the stylesheets available for the form.

The text for the column headers is specified in the **Header** column of the configuration table.

### Print report after page

This property specifies after which page of the form the report table is inserted when the form is printed.

Clicking on this property opens a drop-down menu which contains the list of the page numbers of the form. When no page is selected, the report table is inserted after the last page of the form.

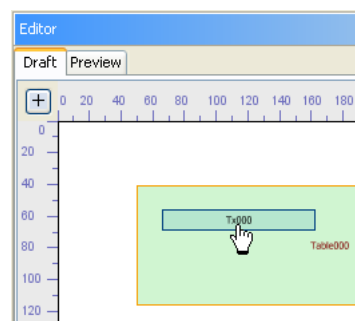
## 4.4.7.2

### Assembling tables

After the table container has been placed, the table has to be assembled. This is done by creating a master row within the table container, and by placing the according form elements. After the master row is assembled, it may be necessary to adjust the table properties (see section 4.4.7.1)

Form elements which are placed within the table container are assigned to it, and become part of the table row. When a form element becomes assigned to a table container, it changes its color (see Fig. 4-137).

**Fig.4-137** *Placing form elements in the table container*



The color change is a visual indication that the form element has become part of the table.

Most form elements may become part of a table row. The following elements cannot be placed into a table container: Line, Rectangle, Image, Infopoint and Password field. With these elements, the cursor changes to a “forbidden” cursor when the mouse is moved over a table container (see Fig. 4-91). Barcode fields may be placed in a table container, but are shown only in the first row.

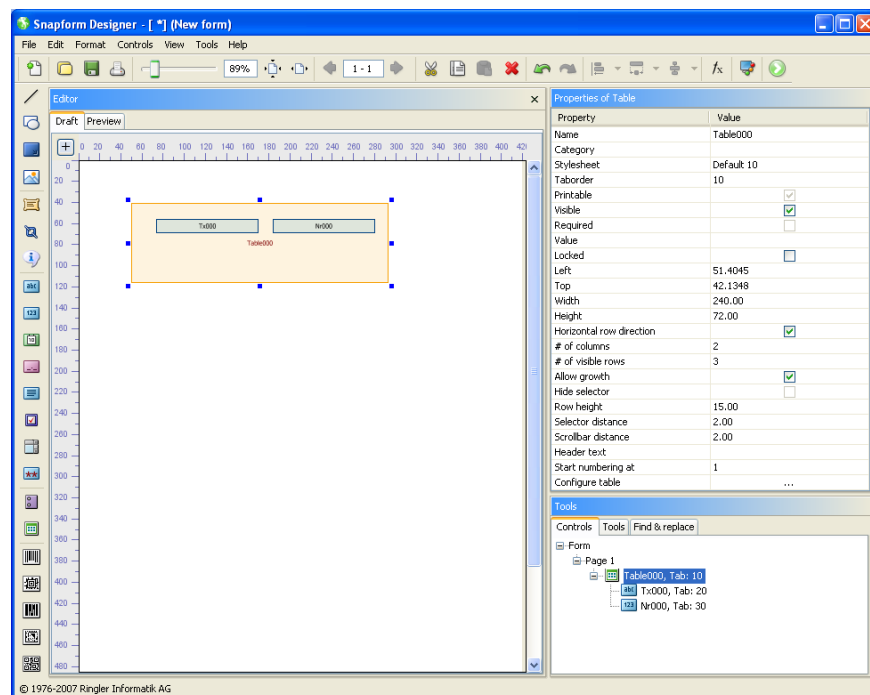
**Attention:** *Datamatrix barcode fields may be placed in a table container, but may lead to a serious corruption of the form.*

**Fig.4-138** “Forbidden” cursor with form elements which cannot be placed in a table container



This step is repeated until the master row is completed (see Fig. 4-139).

**Fig.4-139** Table container with placed form elements



The document structure shows in the example the association of the form elements **Tx000** and **Dt000** to the table container **Table000**.

Form elements which were present in the form before the table container was placed, cannot be added to the group of table elements. It is not possible to add form elements which have first been placed outside of the table container into the table container.

Field elements belonging to a table container must be placed within its perimeter, in order to be displayed properly. They can not be moved outside of the table container.

When a table container is moved or scaled, the associated elements retain their relative position. This means that it is sufficient to move the table container when a table has to be placed elsewhere on the page.

**Helpful Hint:** *As every element placed within a table container (no matter whether intentional or not) is added to the table, it may be recommended to lock the table container (**Locked** property checked) for the on-going development work with the form.*

After creating the master row (which may consist of more than one visual row), the table must be configured.

The following steps are intended for tables consisting of multiple rows ("phantom tables" which consist of one single row are discussed below):

**Helpful Hint:** *It is recommended to regularly verify the results of the actions described below in the Preview mode.*

1. Adjust the value of **Row height** to the actually needed height for a row.

The value of Row height must be greater than the height of the elements of the logical row, in order to prevent overlapping elements when the table is displayed.

This is easily done by selecting all the elements within the table container, copying and pasting them again. Assemble the still active

selection with **Edit —> Group** to a group. The property window of the group displays information about its dimensions.

**Helpful Hint:** *It is possible to keep open a second Snapform Designer window with an empty form, and to paste the copied elements into this window. This will keep the actual form free of superfluous elements.*

2. Adjust the height of the table container, **Row height** and **# of visible rows**.

The height of the table container must be greater than the number of visible rows times the height of the row, so that the last row of the table will not appear cropped. On the other hand, it should not be considerably greater, as this would lead to excessive whitespace, and a scrollbar would be displayed in a confusing way.

3. Controlling the selector

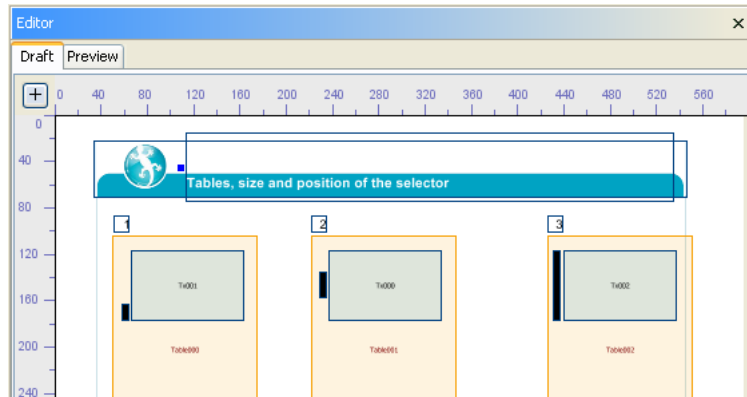
The selector is a visual element of the table which is displayed only in Preview mode and in the Snapform Viewer. It is used to indicate the row number of the table, as well as to select the active row. The size of the selector corresponds to the height of the table element being part of the first (the leftmost) column.

The first parameter of the selector which can be set is its distance from the table container (**Selector distance**). Its default value of 2 Pt is common and generally usable.

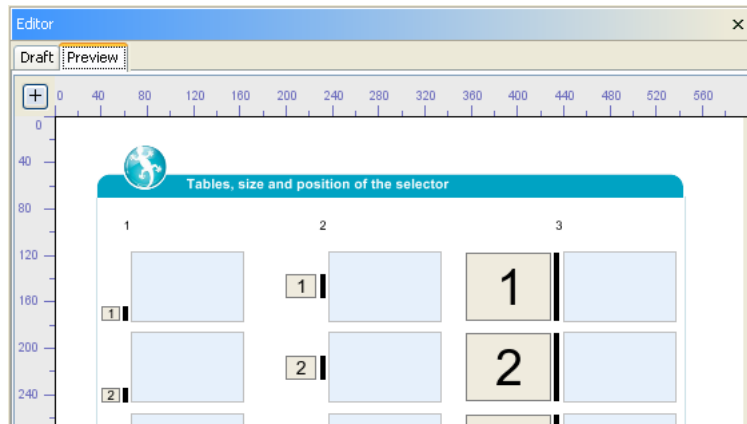
The second parameter for the selector is the number of the first row in the table (**Start numbering at**). Normally, this value is 1, so that the rows are numbered in ascending order, beginning with 1. This value can, however, be set to any integer.

The size and relative position to the row can be controlled by placing a very narrow label with the requested height and position of the selector as leftmost element in the table container, as it is shown in the sample document *table\_selectors.qdf* (see Fig. 4-140 (Draft mode) and Fig. 4-141 (Preview mode)).

**Fig. 4-140** *Size and position of the selector in Draft mode*



**Fig. 4-141** *Size and position of the selector in Preview mode*



- 1 Table 000: Label Lb000, Height 13 Pt, Position aligned with bottom border of text field Tx001 (Height 60 Pt)
- 2 Table001: Label Lb001, Height 21.7 Pt, Position centered with text field Tx000 (Height 60 Pt)
- 3 Table 002: Label Lb002, Height 60 Pt, Position aligned with top border of text field Tx002 (Height 60 Pt)

This example shows three otherwise identically defined tables with different height and position of the first (leftmost) row element (a narrow label which has been colored black in order to be more visible). This allows an extensive control of the selector.



The following steps are only necessary when the **Allow growth** property is active.

#### 4. Setting the position of the scrollbar

The scrollbar appears outside of the table container at its right border. The default value of the **Scrollbar distance** property of 2 Pt has been a good value based on experience. If necessary, the height and width of the table container must be adjusted to prevent the scrollbar from appearing too detached from the table.

The appearance of the scrollbar is controlled from the operating system, and cannot be controlled from Snapform.

#### 5. Specifying the detail report

When the table contains more rows than can be displayed, an additional page is created when the document is printed. This additional page is called "report". This report is a very simply designed table which must be configured.

The report is only displayed in the Print preview. As the Snapform Designer cannot create a Print preview, the form must be saved and then displayed in the Snapform Viewer. At this time, the Print preview can be created by clicking on the Print preview tool. Moving to the following pages will display the report.

The first property of the report is the title which is entered in the **Header text** property.

Further entries are in the **Configure table** window (see Fig. 4-136).

In the report specification table, the values for the column width should be verified. In an unmodified table, the values of the width of the column fields are used. This means that the columns of the report are proportionally the same width as the columns in the form. It may be useful in certain cases to modify the column widths in the report (for example the width of the label inserted to control the selector).

In any case, column headers should be entered. While these headers are present in the form, they must be explicitly specified for the report in the **Header** column.

The **Show total** check box may be checked. In this case, and if the field type of the column allows it, the total of the values of the column is calculated and displayed below the report table. Whether this is desirable depends on the actual use of the table.

The **Column for message** should, whenever possible, be specified. Otherwise, there will be no reference to the report page in the form. Note that it does not matter for the Print preview if the field of the column is able to display the “see listing” text (meaning that date or number fields do work). The field must be wide enough to display the whole text. The font type and size used for the text is controlled via the stylesheet used for the column field.

For the report title and the column header, a suitable stylesheet must be selected (**Style for report title**, and **Style for report headers**). The Default stylesheet is applied by default.

The report will be added, if not specified otherwise, after the last page of the form. When it should appear at another place in the form, select the page after which it should appear in the **Print report after page** property.

Finally, specify by clicking on the according radio button the page orientation; whether the page should be printed in portrait or landscape format.

The row height in the report depends on the value of the **Row height** in the table, which will be scaled proportionally to the table width. This ensures that the report can display at least the same amount of information as the table in the form.

When the table is a “phantom table”, only one single row is displayed in the form. This means that the configuration of the table is particularly important. A “phantom table” is configured as follows:

**Helpful Hint:** *It is recommended to regularly verify the results of the actions described below in the Preview mode.*

1. Adjust the value of **Row height** to the actually needed height for a row.

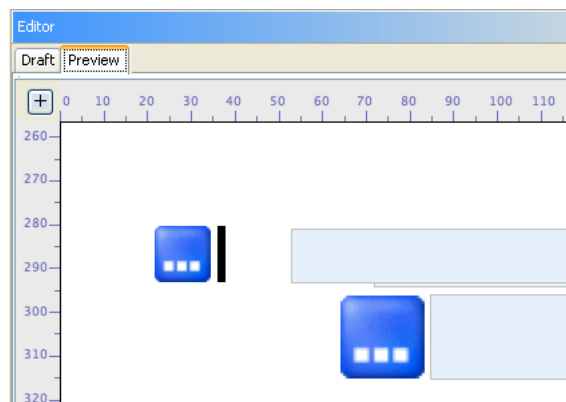
The value of Row height must be greater than the height of the elements of the logical row, in order to prevent overlapping elements when the table is displayed.

As only one single row is displayed, the value for the row height should be in the range of the height of the table container.

2. Activate buttons to open the data entry table

As the table displays only one single row, the capabilities for entering data are limited. This is why the actual data entry and screen display occurs in the data entry table. This table is opened by clicking on a button (see Fig. 4-142).

**Fig.4-142** *Button to open the data entry table (zoom level very high)*



This button gets activated when in the **Configure table** dialog, one or several **Show total** boxes are checked.

The button appears to the left of the specified field element. The size depends on the height of that element, as the example in Fig. 4-142 shows.

3. Specifying the data entry table

The data entry table is screen-oriented, and it cannot be printed (its equivalent is the report). It can be specified to some extent. Certain properties are used by the data entry table and the report.

The first property of the data entry table is the title which is entered in the Header text **property**.

Further entries are in the **Configure table** window (see Fig. 4-136).

In the report specification table, the values for the column width should be verified. In an unmodified table, the values of the width of the column fields are used. This means that the columns of the data entry table are proportionally the same width as the columns in the form. It may be useful in certain cases to modify the column widths in the report (for example the width of labels).

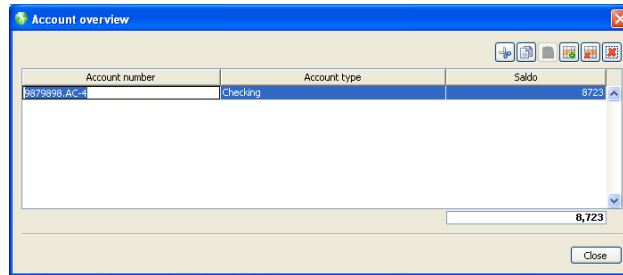
In tables with multiline logical rows, it is important that the numbering of the columns occurs strictly from left to right, and does not consider the visual row of the according field. It is important to define column headers properly (see below).

In any case, column headers should be entered. While these headers are present in the form, they must be explicitly specified for the data entry table in the **Header** column.

The **Show total** check box may be checked. This means that the button to open the data entry table will be displayed, and that in the data entry table, the current column total for the according table is shown. This total is also displayed in the report. The total is only displayed if the field type of the column allows it.

With this, the data entry table is specified. In Preview mode, the button to open this table appears. The example shown in Fig. 4-143 comes from the sample file *tables.qdf*.

**Fig.4-143** *Data entry table  
for item 3 of  
tables.qdf*



Data entry into the table is done like entering data into a typical spreadsheet.

The data entry table has a toolbar consisting of 6 tools (see Fig. 4-144).

**Fig.4-144** *Toolbar of the data  
entry table*



#### Cut

The Cut tool removes the current row and puts it in the clipboard. The subsequent rows move up one position.

#### Copy

The Copy tool copies the current row in the clipboard.

#### Paste

The Paste tool inserts the row stored in the clipboard at the end of the table.

#### Add row

The Add row tool inserts a new empty row above the active row.

#### Delete row

The Delete row tool deletes the active row from the table.

#### Delete

The Delete table tool deletes the contents of the data entry table and removes all the rows.

### 4. Specifying the detail report

In phantom tables, an additional page is created when the document is printed. This additional page is called "report". This

report is a very simply designed table which must be configured. Certain properties of this table are used in common with the data entry table and do not need to be configured any further.

The report is only displayed in the Print preview. As the Snapform Designer cannot create a Print preview, the form must be saved and then displayed in the Snapform Viewer. At this time, the Print preview can be created by clicking on the Print preview tool. Moving to the following pages will display the report.

The title of the report has already been specified for the data entry table in the **Header text** property.

Further entries are in the Configure table window (see Fig. 4-136).

The column widths for the report are identical to the ones of the data entry table, and have been defined in that step. The column headers have been defined in the **Header** column, and the choice of columns with a total has been specified with the **Show total** check box.

The **Column for message** property should, if possible, be specified, as otherwise, there is no reference in the form to the report page. It doesn't matter for the Print preview whether the according column is able to display the text "see listing" (which means that date or number fields do work). The field must, however, be wide enough to display the whole text. The font type and size used for the text is controlled with the stylesheet of the column field.

For the report title and the column header, a suitable stylesheet must be selected (**Style for report title**, and **Style for report headers**). The Default stylesheet is applied by default.

The report will be added, if not specified otherwise, after the last page of the form. When it should appear at another place in the form, select the page after which it should appear in the **Print report after page** property.

Finally, specify by clicking on the according radio button the page orientation; whether the page should be printed in portrait or landscape format.

The row height in the report depends on the value of the **Row height** in the table, which will be scaled proportionally to the table width. This ensures that the report can display at least the same amount of information as the table in the form.

With this, the table is specified, and the form can be used.

### 4.4.7.3

### Examples of tables

The following examples show the different variations of tables in Snapform.

The document *tables.qdf* (see Fig. 4-145 (Draft mode) and Fig. 4-146 (shown in the Snapform Viewer)) shows various examples of tables.

After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties.

**Fig. 4-145** Table examples  
from tables.qdf, in  
Draft mode

The screenshot shows the Snapform Designer Editor in Draft mode. The interface includes a ruler at the top and left. The main area displays four table examples:

- Table000:** A table with columns: Storage Unit, In, Out, Units. It has a total row labeled 'Total000'.
- Table001:** A table with columns: Name, First name, Date of birth, Membership. It has a total row labeled 'Total001'.
- Table002:** A table with columns: Account number, Account type, Balance. It has a total row labeled 'Total002'.
- Table003:** A table with columns: Number, Item, Unit price, Units, Amount. It has a total row labeled 'Total003'.

On the right side, there is a list of flight codes labeled 'Table004':

Flight	Code
Fit 1	Tx006
Fit 2	Tx007
ZRH	D001
FRA	D002
MUC	D003
LHR	D004
IAD	D005
ORD	D006
BOS	D007
SFO	D008

**Fig. 4-146** Examples from  
tables.qdf in the  
Snapform Viewer,  
not filled out

The screenshot shows the Snapform Viewer displaying the same four table examples as in Fig. 4-145, but they are not filled out. The flight codes list (Table004) is also present on the right side.

The tables are:

- Table000:** Storage Unit, In, Out, Units. Total row: Total000.
- Table001:** Name, First name, Date of birth, Membership. Total row: Total001.
- Table002:** Account number, Account type, Balance. Total row: Total002.
- Table003:** Number, Item, Unit price, Units, Amount. Total row: Total003.

The flight codes list (Table004) is:

Flight	Code	
Flights 1	Flights 2	Flights 3
Fit 1		
Fit 2		
ZRH		
FRA		
MUC		
LHR		
IAD		
ORD		
BOS		
SFO		



- 1 Normal table, fixed number of rows
- 2 Table with variable number of rows, and 3 displayable rows
- 3 Phantom table with one single displayed row
- 4 Table with variable number of rows, several visual rows per logical row, Totals below the table
- 5 Table with variable number of rows, vertical logical rows

In Fig. 4-147, the same form is shown, but with filled out table fields

**Fig. 4-147** Examples from *tables.qdf* in the Snapform Viewer, filled out

The screenshot shows a Snapform Viewer window titled 'tabellen.qdf \* - Snapform Viewer' at 159% zoom. The form contains five examples of tables:

**Example 1: Storage Unit**

Storage Unit	In	Out
1 First floor	23,232 Units	19,823
2 Second floor	3,432 Units	2,532
3 Shed	44 Units	27

**Example 2: Name**

Name	First name	Date of birth	Membership
3 Tell	William	18.10.1263	<input checked="" type="checkbox"/> Yes <input type="checkbox"/> No
4 Dupont	René	05.04.2003	<input type="checkbox"/> Yes <input checked="" type="checkbox"/> No
5 Dupond	Roland	05.04.2003	<input type="checkbox"/> Yes <input checked="" type="checkbox"/> No

**Example 3: Account number**

Account number	Account type	Balance
<a href="#">See listing</a>		35,735

**Example 4: Number Item**

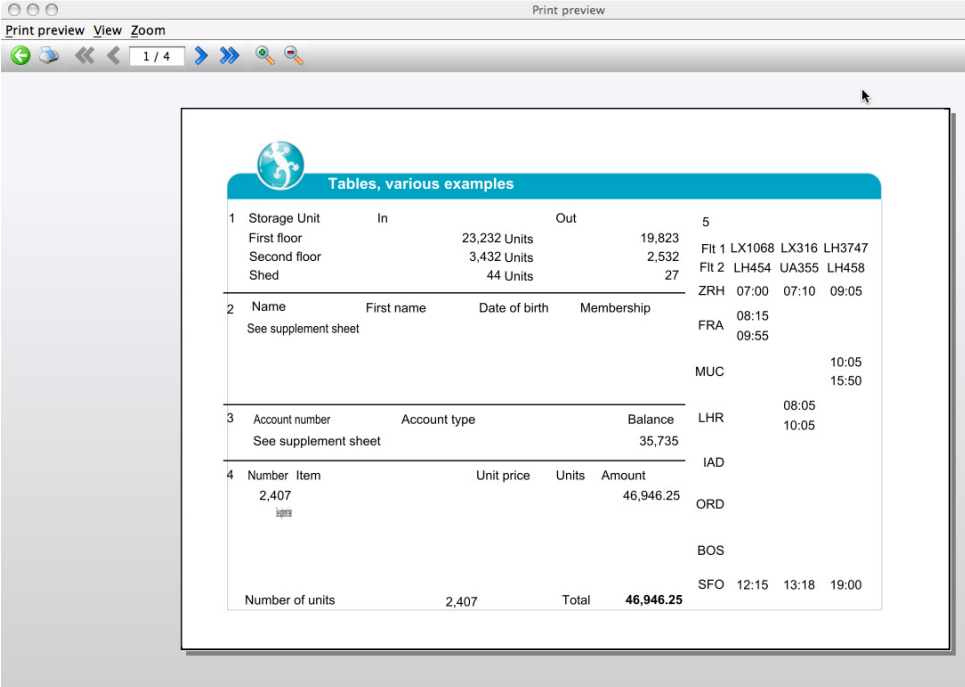
Number	Item	Unit price	Units	Amount
2	45 yellow parrots in stock	at 56,234 kg		2,530.55
3	2,342 rusty nails in stock	at 3.45 kg		8,079.90
4	8 ihavethe-blue in stock	at 4,523 Liter		36,184.00
Number of units				2,407
Total				46,946.25

**Example 5: Flights**

	Flights 1	Flights 2	Flights 3
Fit 1	LX1068	LX316	LH3747
Fit 2	LH454	UA355	LH458
ZRH	07:00	07:10	09:05
FRA	08:15		
	09:55		
MUC			10:05
			15:50
LHR			08:05
			10:05
IAD			
ORD			
BOS			
SFO	12:15	13:18	19:00

In Fig. 4-148, the same form is shown in the Print preview.

**Fig. 4-148** Examples from *tables.qdf* in the Snapform Viewer, Print preview



Print preview

Print preview View Zoom

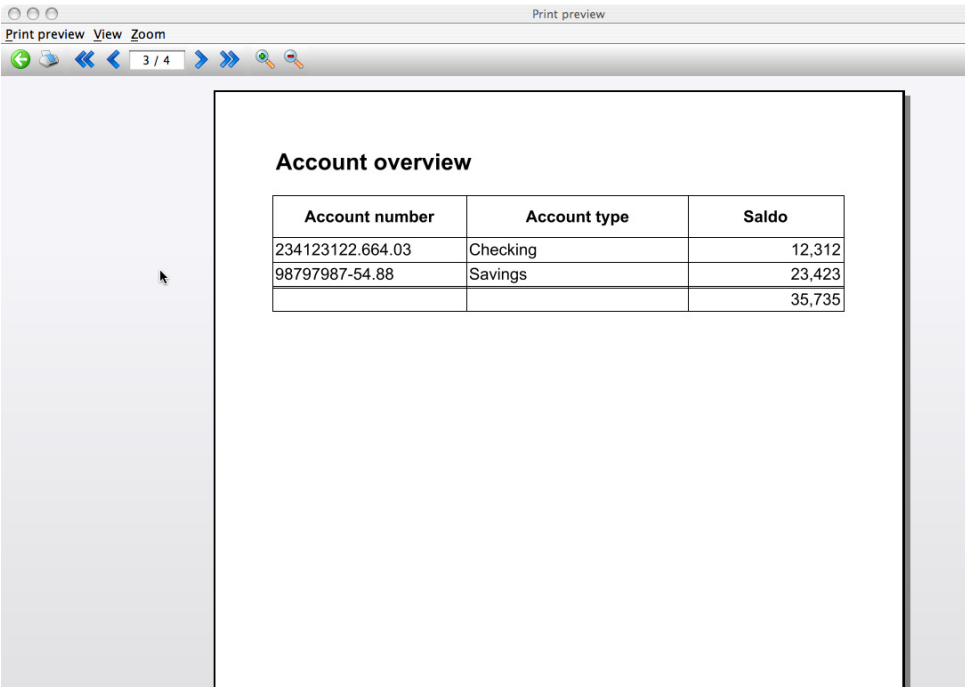
1 / 4

**Tables, various examples**

1	Storage Unit	In	Out	5
	First floor	23,232 Units	19,823	Fit 1 LX1068 LX316 LH3747
	Second floor	3,432 Units	2,532	Fit 2 LH454 UA355 LH458
	Shed	44 Units	27	ZRH 07:00 07:10 09:05
2	Name	First name	Date of birth	Membership
	See supplement sheet			FRA 08:15 09:55
				MUC 10:05 15:50
3	Account number	Account type	Balance	LHR 08:05 10:05
	See supplement sheet		35,735	IAD
4	Number	Item	Unit price	Units
	2,407			Amount 46,946.25
				ORD
				BOS
				SFO 12:15 13:18 19:00
	Number of units	2,407	Total	46,946.25

In Fig. 4-148, finally, the report for item 3 is shown.

**Fig. 4-149** Report of item 3 of *tables.qdf* in the Snapform Viewer



Print preview

Print preview View Zoom

3 / 4

**Account overview**

Account number	Account type	Saldo
234123122.664.03	Checking	12,312
98797987-54.88	Savings	23,423
		35,735

## 4.4.8

## Active elements

Active elements are elements of the form which are used as trigger for user actions, but otherwise do not have any further significance for the form. Typical active elements are buttons or links.


Snapform has two types of active elements which can be placed in a form: Hyperlinks and Infopoints. In addition, a set of command buttons in the Snapform Viewer can be activated through the **Form settings** properties of the form.

### 4.4.8.1

### Hyperlinks

Hyperlinks are form elements which take the user to another place. This may be either another form element, or an Internet address may be called and be displayed using specific software (web browser, mail client).

Hyperlinks can be used to create an extensive reference and help system, where information available in the Intranet or Internet can be referred to.

The Hyperlink tool  creates an active area which is used as the starting point for the hyperlink. After selecting the tool, a cursor with a hyperlink in default size appears (see Fig. 4-150).

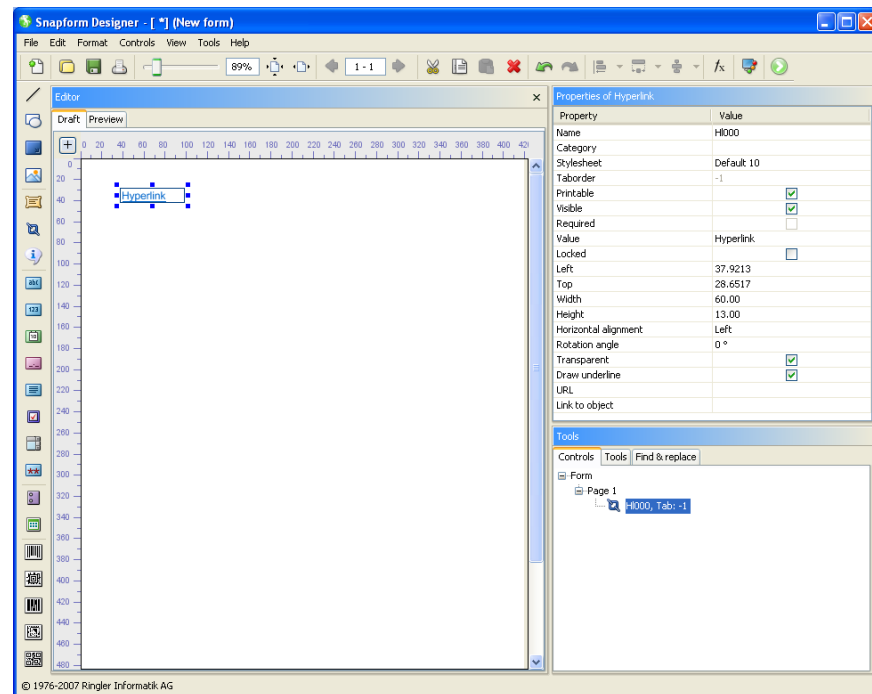
**Fig. 4-150** *Hyperlink cursor*



Assigned to the Hyperlink tool is a hyperlink field, 60 pt wide and 13 pt high. Clicking the mouse places it on the workspace. It is automatically assigned a name, consisting of the characters **HI** and a three-digit consecutive number, beginning with **000**. The first hyperlink placed in a form has therefore the name **HI000**. In the object structure it is added to the according page. In Fig. 4-151 the first hyperlink of the form has been placed and then selected.

In order to mark and as a reminder to configure, a newly placed hyperlink has the text **Hyperlink** which covers the field name.

**Fig. 4-151** Newly created hyperlink



The length and the width of the hyperlink can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the hyperlink are controlled in the Properties window (see Fig. 4-152).

**Fig. 4-152** *Properties of a hyperlink*

Properties of Hyperlink	
Property	Value
Name	Hl000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input checked="" type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	Hyperlink
Locked	<input type="checkbox"/>
Left	37.9213
Top	28.6517
Width	60.00
Height	13.00
Horizontal alignment	Left
Rotation angle	0 °
Transparent	<input checked="" type="checkbox"/>
Draw underline	<input checked="" type="checkbox"/>
URL	
Link to object	

## Name

The name of the hyperlink in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For hyperlinks, the zones **Font settings** and **Supplemental font support** in the **General**

**settings** tab as well as the **Infopoint and hyperlinks** tab of the stylesheet editor are of importance.

### **Taborder**

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence.

### **Printable**

When the box is checked, the hyperlink (its text) will be printed.

### **Visible**

When the box is checked, the hyperlink is displayed on screen.

### **Required**

This property is always deactivated, and cannot be changed.

### **Value**

This is the text displayed in the hyperlink. The default value is Hyperlink.

Clicking on this property opens the expression editor for the text entry. This may either be a constant text (as the default value), or the result of a calculation (which allows to assemble dynamic URLs).

Only simple text is supported; HTML code is displayed literally.

### **Locked**

When this box is checked, the hyperlink gets locked in the Snapform Designer, and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

### **Left**

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the hyperlink.

### **Top**

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the hyperlink.

<b>Width</b>	Width of the bounding box of the hyperlink.
<b>Height</b>	Height of the bounding box of the hyperlink.
<b>Read-only</b>	When this box is checked, the hyperlink is marked as read-only, and it cannot be activated. The target may still be changed as the result of an expression. A read-only hyperlink is still visible to expressions, and its assigned expressions are evaluated according to its state.
<b>Horizontal alignment</b>	<p>This property defines how the text is horizontally aligned in the hyperlink.</p> <p>When the property field is clicked, a drop-down menu appears with the options <b>left</b>, <b>center</b>, <b>right</b>, which places the text left-aligned, centered or right-aligned within the hyperlink field. Default is <b>left</b>.</p>
<b>Rotation angle</b>	The text in the hyperlink field can be rotated by multiples of 90°. Clicking on this property opens a drop-down menu with the selection of the rotation angles <b>0°</b> , <b>90°</b> , <b>180°</b> , <b>270°</b> . The default value is <b>0°</b> .
<b>Transparent</b>	The background of the hyperlink is set to transparent (no covering background color) with this check box. This property is active by default.
<b>Draw underline</b>	Hyperlinks in the web environment are usually underlined. Checking this box underlines the text of the hyperlink field. This property is active by default.
<b>URL</b>	<p>This is the address (URL) to which the hyperlink leads. Snapform supports all the URI protocol which are registered in the host computer. The following protocols are typically supported:</p> <ul style="list-style-type: none"><li>• http:, https: web pages</li><li>• mailto: E-mail</li></ul>

- file: open file

**Note:** *It is in the sole responsibility of the creator of the form to make sure that the target address is correct and without risks for the user. Ringler Informatik AG cannot be made liable for any damage caused by the use of hyperlinks.*

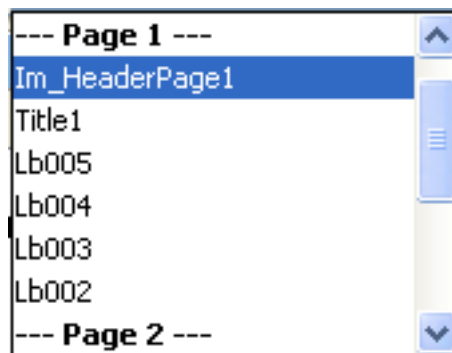
This property is a simple entry field into which the URL can be entered.

## Link to object

Besides links to external places (which are controlled with the **URL** property), the hyperlink tool also offers the possibility to create links within the form. This property is used for such internal links.

When this property is selected, a drop-down list containing all the elements of the form layer (all elements with a taborder value greater than 0, including Image fields and labels). The elements are grouped to the document's pages, which means that elements with multiple occurrences can be individually targeted (see Fig. 4-153). A field can now be selected.

**Fig.4-153** Target object selection in Link to object

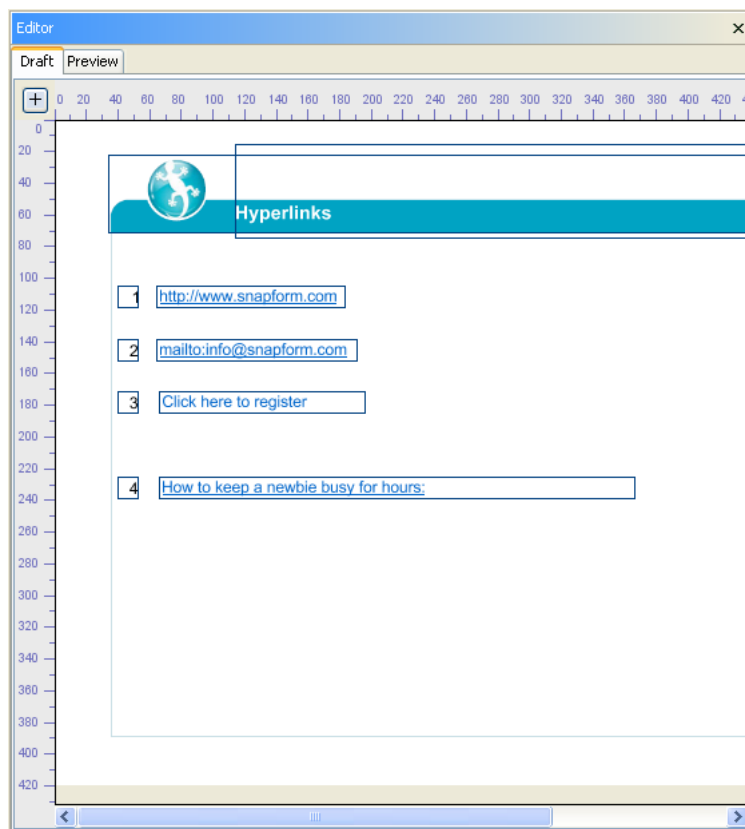


The document *hyperlinks.qdf* (see Fig. 4-154 (Draft mode) and Fig. 4-155 (shown in the Snapform Viewer)) shows various examples of hyperlinks.

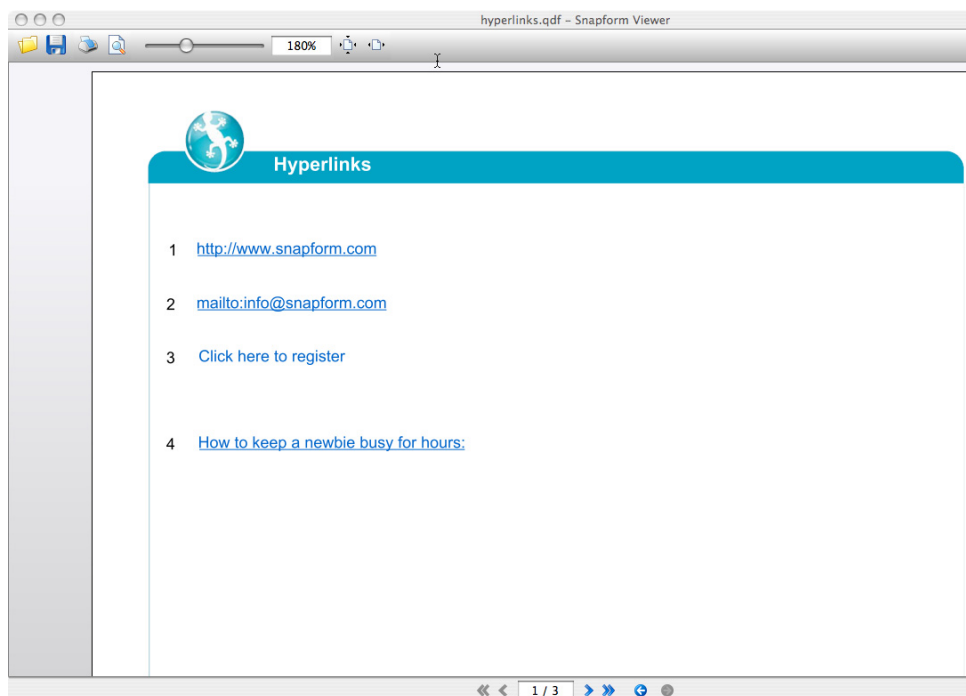
After selecting the Field elements in the Snapform Designer, the Properties window displays information about the according properties.



**Fig. 4-154** Table examples  
from  
*hyperlinks.qdf*, in  
Draft mode



**Fig. 4-155** Examples from  
*hyperlinks.qdf* in  
the Snapform  
Viewer



- 1 HI000: Simple hyperlink to a web page; URL: **http://www.snapform.com**
- 2 HI001: Hyperlink to mail client; URL: **mailto:info@snapform.com**
- 3 HI002: Hyperlink to mail client with pre-filling of Subject and Body; underlining unchecked; URL: **mailto:info@snapform.com?subject=Registration&body=I%20want%20to%20register**
- 4 HI003: Internal hyperlink to label Lb000, which is on the second page of the form (selection in **Link to object** see Fig. 4-153).


#### 4.4.8.2

### Infopoints

A great advantage of electronic forms is that additional information and explanations can be made available and displayed very easily at the place where they are needed (without overloading the form). One possibility to display explanations is the mousetip help for entry fields. This is very useful, but it should only contain field-related information.

In addition, there are hyperlinks which refer to external sources (or an accordingly created page of the document itself).

For situations where additional information is not directly connected with a field, and you don't want the user to be drawn too far away from the page of the form, Snapform has so-called Infopoints. An infopoint is an active element where text appears when the user moves the mouse over it. An infopoint can be built up in a way that it serves an area of the form. The active element itself, however, is small enough to not obstruct the form.

The Infopoint tool  creates a field where passing the mouse over it, opens a text window. After selecting the tool, a cursor with an Infopoint in default size appears (see Fig. 4-156).

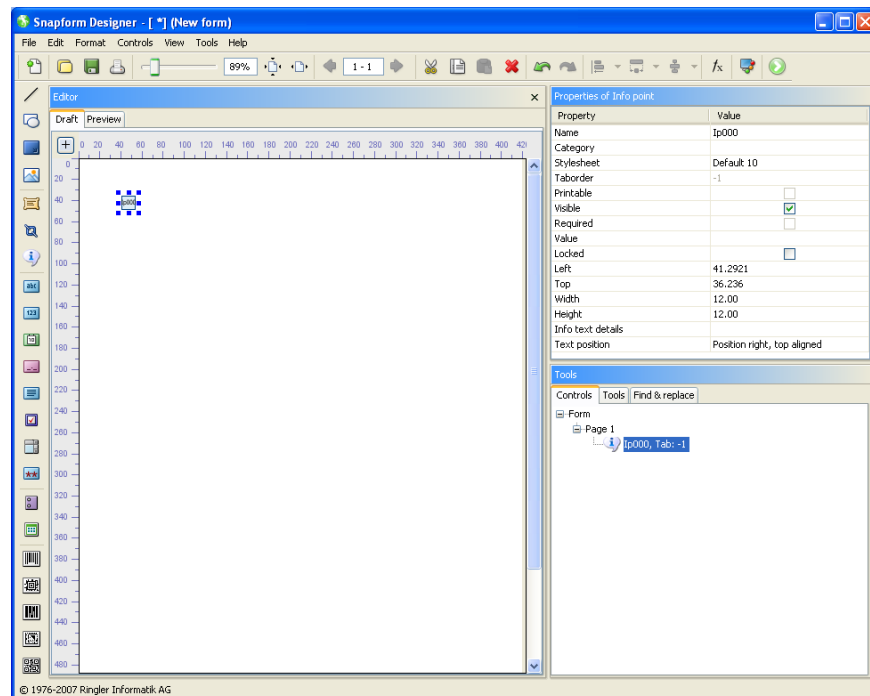
**Fig. 4-156** *Infopoint tool  
cursor*



Assigned to the Infopoint tool is an infopoint field, 12 pt wide and 12 pt high. Clicking the mouse places it on the workspace. It is automatically

assigned a name, consisting of the characters **lp** and a three-digit consecutive number, beginning with **000**. The first infopoint field placed in a form has therefore the name **lp000**. In the object structure it is added to the according page. In Fig. 4-157 the first infopoint of the form has been placed and then selected.

**Fig.4-157** Newly created infopoint



The length and the width of the infopoint can be changed by dragging the anchor points of the bounding box. The position of the rectangle can also be changed by "grabbing" and dragging. When passing an anchor point, the cursor changes to double-arrows which indicate the moving direction of the anchor point. When "grabbing" and pressing the mouse button, the cursor changes to an outlined arrow.

The other features of the infopoint are controlled in the Properties window (see Fig. 4-158).

**Fig. 4-158** *Properties of an infopoint*

Properties of Info point	
Property	Value
Name	Ip000
Category	
Stylesheet	Default 10
Taborder	-1
Printable	<input type="checkbox"/>
Visible	<input checked="" type="checkbox"/>
Required	<input type="checkbox"/>
Value	
Locked	<input type="checkbox"/>
Left	41.2921
Top	36.236
Width	12.00
Height	12.00
Info text details	
Text position	Position right, top aligned

## Name

Name of the infopoint field in the document. This name can be changed.

## Category

Assignment to one or several categories which allows you to build groups of elements. Clicking this field opens the Categories dialog (see Fig. 4-23).

The categories are selected by clicking on the according check box. With **OK** the selection becomes valid.

## Stylesheet

Stylesheet valid for the element. When there is no explicit stylesheet selection, the first default stylesheet is used. For infopoints the zones **Font settings** and **Supplemental font support** in the **General settings** tab as well as the **Infopoint and hyperlinks** tab of the stylesheet editor are important.

**Helpful Hint:** *As infopoint fields are often created with white text on dark background, the readability of the text (**Value** property) can be improved by*

*using a stylesheet with a bold font (such as **Default 10 Bold** instead of **Default 10**).*

## Taborder

Sequence number of the element in the tab order. This value is **-1** and cannot be changed. This means that the element is not part of the tabbing sequence.

## Printable

This property is always deactivated, and cannot be changed.

## Visible

When the box is checked, the infopoint is displayed on screen.

## Required

This property is always deactivated, and cannot be changed.

## Value

In this property of the infopoint field, the text is entered which appears in the infopoint field, and which indicates that this field is an infopoint. Typical values are **?** or **...** or **i**. Clicking this property opens the expression editor for text entry.

The text of an infopoint field must be simple text; HTML code is not interpreted and will be displayed literally.

When using an expression like

```
= "?|http://www.snapform.com"
```

it is possible to display a complete HTML page without the need for an entry in the **Info text details** property. This page must, however, be pure HTML 3.2-compatible code. Otherwise, unexpected text may be displayed.

## Locked

When this box is checked, the infopoint field gets locked in the Snapform Designer and cannot be (accidentally) moved. In addition, all other properties in the Properties dialog are made inactive. Access to these properties is only available after deactivating this check box.

## Left

X-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the infopoint.

## Top

Y-coordinate of the upper left corner of the bounding box of the element, measured in Points. This property allows precise numeric positioning of the infopoint.

## Width

Width of the bounding box of the infopoint.

## Height

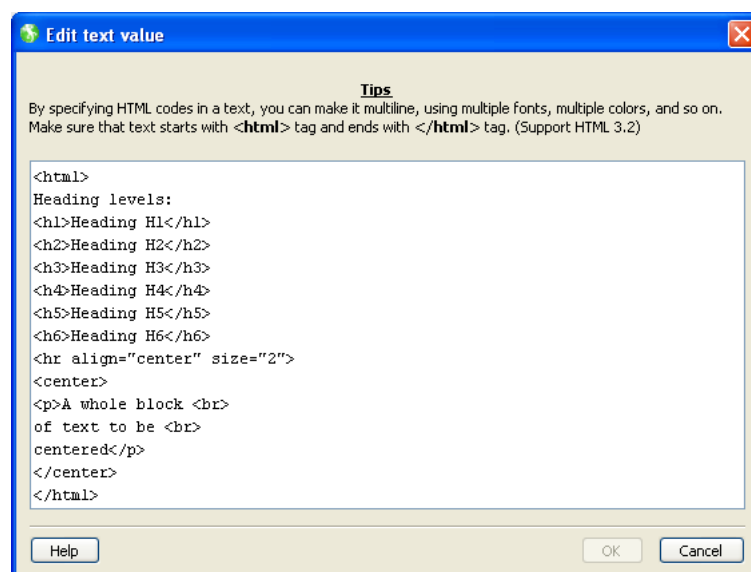
Height of the bounding box of the infopoint.

## Info text details

This property specifies the text which gets displayed when the mouse cursor is moved over the infopoint. This text can be either simple unformatted text, or it can be formatted text in HTML form.

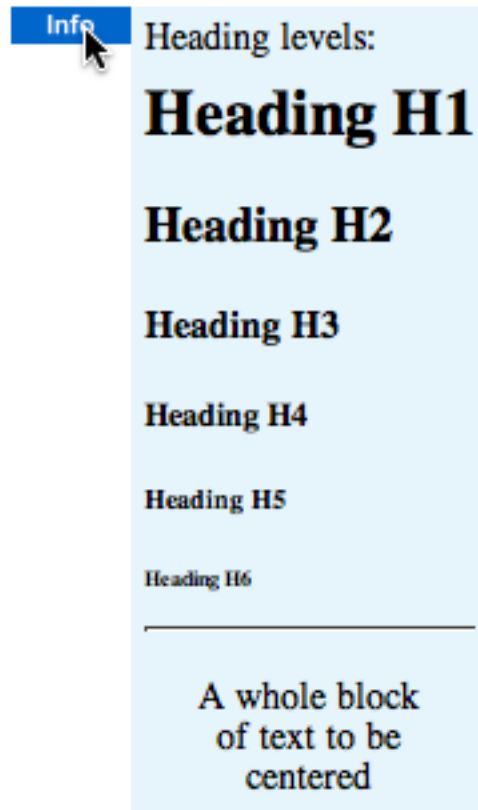
When this property is selected, a ... button appears at the right side of the property field. Clicking this button, or double-clicking the property field opens the text editing window (see Fig. 4-159). This window is identical to the short text editing window of entry fields (see also section 4.4.2.1). As the explanatory text states, it is possible to enter HTML code as well.

**Fig.4-159** *Info text editing window*



The text defined as HTML in Fig. 4-159 appears in the Snapform Viewer as shown in Fig. 4-160.

**Fig.4-160** Infopoint with text defined according to Fig. 4-159



Simple text is displayed as it has been entered, including line breaks. The font type and size depends on the selected stylesheet.

HTML-encoded text is displayed as specified in its tags. The width of the displayed text area depends either on the information from the HTML code, or, if no such information is available, on the length of the longest resulting text. In order to keep the width of the text area within reasonable limits, it is recommended to limit the width (by putting the text into a table cell and specifying the width of the table). It is also possible to force line breaks using the **<BR>** Tag.

The size of the displayed info text is independent of the zoom factor of the viewer. That means that it is larger relative to the visible form with

low zoom factors, and gets smaller with high zoom factors. This feature must be taken into consideration when designing the text.

The supported HTML code corresponds to HTML 3.2 with a few additions from HTML 4.0.1. An overview over the supported tags can be found in section 4.3.6.2.

**Note:** *It is possible to insert referenced images into the info text. However, in order to display these images, an internet connection must be active when the form is used.*

**Note:** *When HTML code is entered, make sure that after the `</HTML>` Tag no further characters are present (not even invisible characters such as Space or New line). If this is the case, the HTML code will be displayed literally.*

**Note:** *It is possible that when confirming the info text, a warning appears stating that the tags were not correct. This is due to a somewhat too strict syntax checking done by the Snapform Designer. It is recommended to accept the modifications with **Yes** and then verify in the Preview mode whether the HTML code is properly interpreted. This message is more likely to appear with lists and tables.*

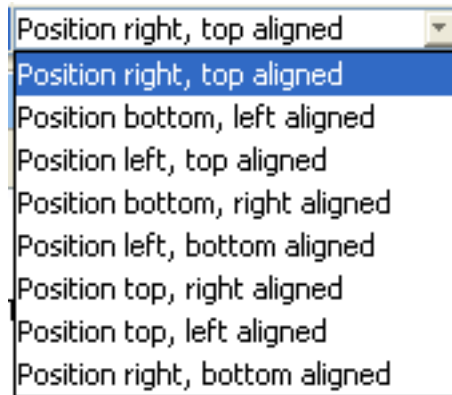
## Text position

This property specifies how the text is displayed in relationship to the infopoint field. This allows good control over which parts of the form are covered by the appearing info text.

Clicking on the property opens a drop-down menu with the selection shown in Fig. 4-161.

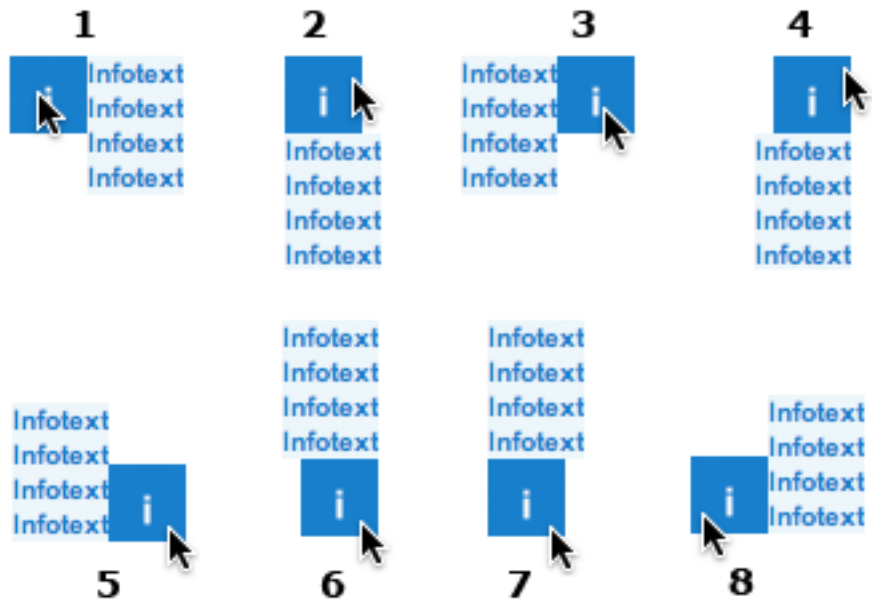


**Fig.4-161** *drop-down menu  
with text positions  
for the infopoint  
field*



“Position” specifies which border of the infopoint field the info text is placed. “Aligned” specifies which border of the infopoint field the info text is aligned. Fig. 4-162 shows the eight possible options.

**Fig.4-162** *Text positions of  
info point fields*



- 1 Position right, top aligned
- 2 Position bottom, left aligned
- 3 Position left, top aligned
- 4 Position bottom, right aligned
- 5 Position left, bottom aligned
- 6 Position top, right aligned
- 7 Position top, left aligned

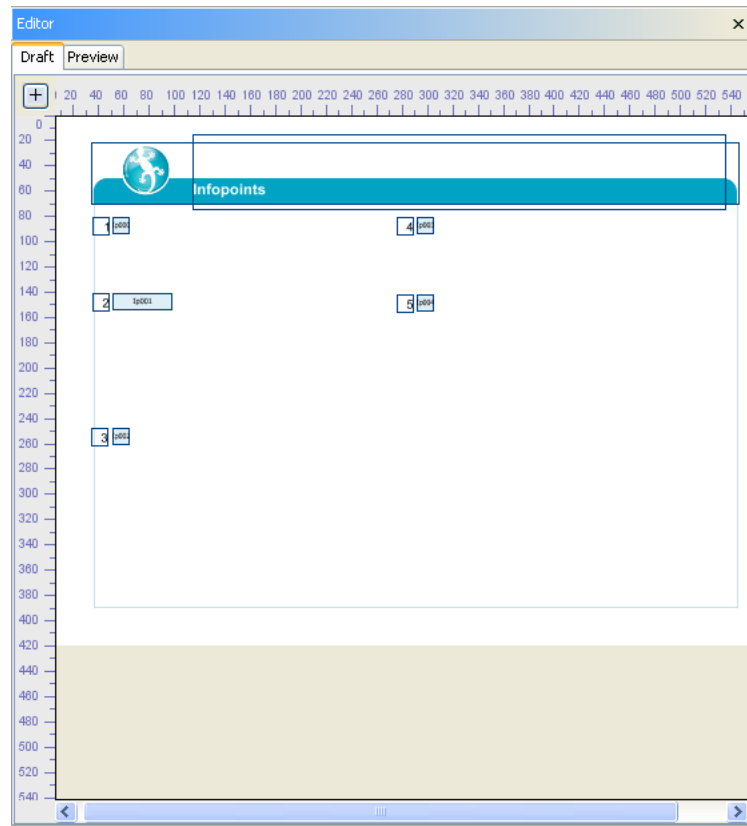
## 8 Position right, bottom aligned

**Note:** *The info text is rendered beyond the border of the form. However, parts outside the document will be cut off. It is recommended that you always verify the info text in the Preview mode of the Snapform Designer, where different assumed zoom factors should be selected.*

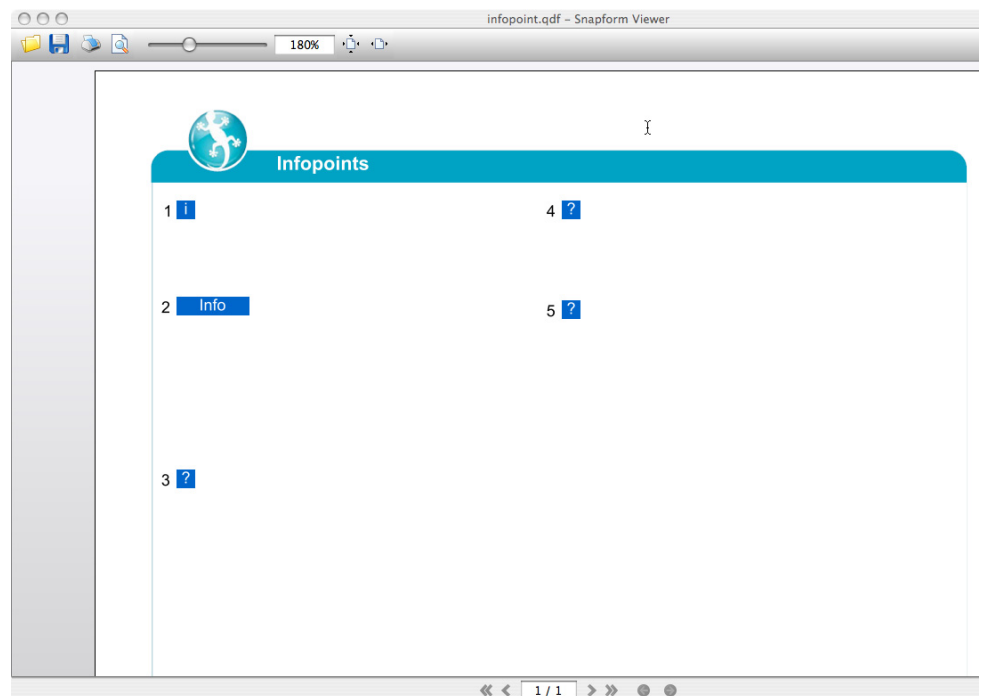
The document *infopoints.qdf* (see Fig. 4-163 (Draft mode) and Fig. 4-164 (shown in the Snapform Viewer)) shows various examples of infopoints. The following picture (see Fig. 4-165) shows the activated info texts (the image has been assembled from various parts using image processing software; it is not possible to open several infopoints at the same time).

After selecting the infopoint field in Snapform Designer, the properties window displays information about the according properties. Double-clicking the field opens the expression editor to display the Text property, and double-clicking the **Info text details** property field shows the info text.

**Fig. 4-163** *Infopoint examples from infopoint.qdf, in Draft mode*



**Fig. 4-164** *Infopoint examples from infopoint.qdf, in the Snapform Viewer*



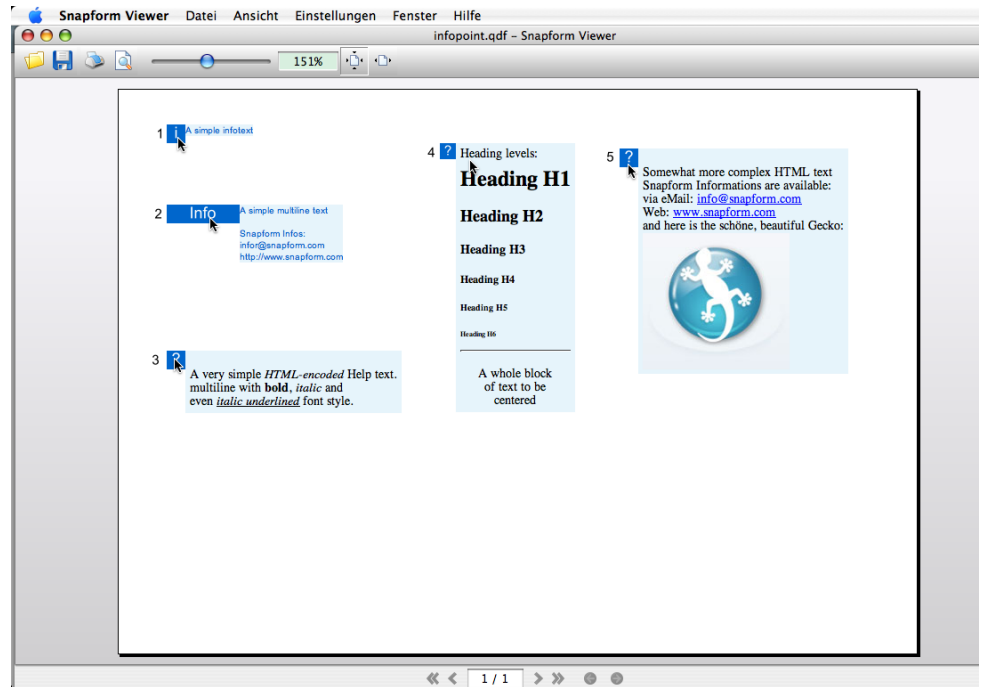
- 1 Ip000: Infopoint, unformatted text, value of **Info text details:**  
**A simple Info text**
- 2 Ip001: Infopoint, unformatted text, value of **Info text details:**  
**A simple multiline text:**  
  
**Snapform Infos:**  
**info@snapform.com**  
**http://www.snapform.com**
- 3 Ip002: Infopoint, HTML-encoded text with in-line tags, value of **Info text details:**  

```
<html>
<p>A simple <em>HTML-encoded</em> Help text.<br>
multiline with <b>bold</b>, <i>italic</i> and<br>
even <i><u>italic underlined</u></i> font.</p>
</html>
```
- 4 Ip003: Infopoint, HTML-encoded text various tags, value of Info text details:  

```
<html>
Heading levels:
<h1>Heading H1</h1>
<h2>Heading H2</h2>
<h3>Heading H3</h3>
<h4>Heading H4</h4>
<h5>Heading H5</h5>
<h6>Heading H6</h6>
<hr align="center" size="2">
<center>
<p>A whole block <br>
of text to be <br>
centered</p>
</center>
</html>
```
- 5 Ip004: Infopoint, HTML-encoded text with links and image, value of **Info text details:**  

```
<HTML>
<P>somewhat more complex HTML text <BR>
Snapform Info are available from <BR>
via E-Mail: <a href=
"mailto:info@snapform.com">info@snapform.com</a><BR>
Web: <a href=
"http://www.snapform.com">www.snapform.com</A><BR>
and here the sch&ouml;lne, beautiful Gecko:<BR>
</P>
</HTML>
```

**Fig. 4-165** Infopoint examples from *infopoint.qdf*, in the Snapform Viewer with opened info texts



#### 4.4.8.3

#### Buttons

Another active element are (push) buttons. These are elements which the user activates (for example by clicking), which initiates an action.

Buttons in the sense of other forms systems are unknown in Snapform. Instead of this, there is a mechanism which activates the most common actions normally assigned to buttons in the Snapform Viewer via specific flags. Estimates say that this, together with internal hyperlinks, handles about 97% of all button actions within a form.

These form actions are specified in the Security settings of the form (see section 4.8.3.3).

#### 4.4.8.4

#### Applied active elements: A help system

An application of active elements is a help system for a form. Help systems are often neglected, but they are worth the effort, particularly with forms that are rather complex, or very frequently used by laypersons. The reduction of user support effort pays very quickly for setting up a good help system.

When setting up a help system, be aware that it may be intended for users which are not familiar with the form, but that it must not slow down or annoy the experienced user. This means that any kind of help must be displayed only as a reaction to an action by the user. Overly active and persistent help will be perceived quite quickly as disturbance.

A help system consists of several levels:

1. direct notes on field level

The first level of the help system should state what has to be entered into the according field. This is an addition to the description of the field on the form itself. The first level should help the user understanding the subject matter required to fill out the form. It is not the task of the first level to explain why a certain entry has to be done, and what it means.

The information on the first level is usually also used by screen readers to guide the user. This must be taken into account when creating help text.

In Snapform, the first level is implemented in the Tooltip text (see section 4.4.2.1).

2. topical notes for groups of fields

The second level of the help system explains logically matching areas of the form. Here, the meaning of the area is discussed. This information is topical and helps the inexperienced user to understand the subject matter. The second level should, on the other hand, not cover data entry specific issues (that has been covered in the first level), unless these entries are explained in a greater context.

In Snapform, the second level is implemented in infopoints (see section 4.4.8.2).

3. extensive topical notes for the whole form

The third level of the help system contains more extensive topics covering the whole form. This information is still stored within the document, but its access is not as immediate. Accessing this information does not need to occur directly, but can be done using a “main switch” which shows and hides a “help layer”.

In Snapform, the third level is implemented with infopoints or an additional page which is accessed either by using hyperlinks or categories which are made visible using a check box. The user still accesses the information via infopoints.

#### 4. background information on form level

The fourth level of the help system covers fundamental background issues about the form, such as legal background, etc. When using this information, the form is left behind. This level will only be used when an extended interest in the topic has to be covered, or when special cases must be treated.

Elements to access the fourth level can either be open and always accessible, or hidden and revealed with the third or maybe second level.

In Snapform, the fourth level is implemented either with hyperlinks (see section 4.4.8.1) or with links embedded in HTML-encoded info texts, see section 4.4.8.2).

## 4.5 Actions and logic

The two biggest advantages of electronic forms versus paper-based forms is that it is possible to add “intelligence” to the form, and to make it interactive. This is normally done using an expression language or a programming language. Snapform uses an expression language which is closely related to the one used in certain spreadsheets.

The difference between expression languages and programming languages reveals itself in the amount of control structures and functions. Expression languages have fewer. On the other hand, an expression language is easier to learn, and expressions can be created using code generators, which simplifies its use even more.

Another difference is that with expressions, field-related results are “pulled in”, whereas in programming languages field-related results also can be “pushed into” a field. The consequence is that for expression languages, the expression must be directly assigned to the field which will show the result.

The expression editor with its user interface is explained in section 3.2.8.

Expressions are processed either from a user action (the user “does something”), or from logic (the form “does something”).

### 4.5.1 Carriers of action and logic

Actions and logic need a carrier, so that they can be executed. In Snapform, such carriers are fields and functions. In older Snapform versions, expressions could only be assigned to labels. This is no longer the case; expressions can now be attached to any field type. It is also possible to define expressions as functions, which has the advantage that the code can be repurposed.



4.5.1.1

Fields

Expressions in fields are “self-referring”. The result is either completely independent from fields (such as showing or hiding pages), or it is written directly into the associated field.

Expressions are specified in the Value property of the according field. Expressions with a calculation always begin with an equal sign.

4.5.1.2

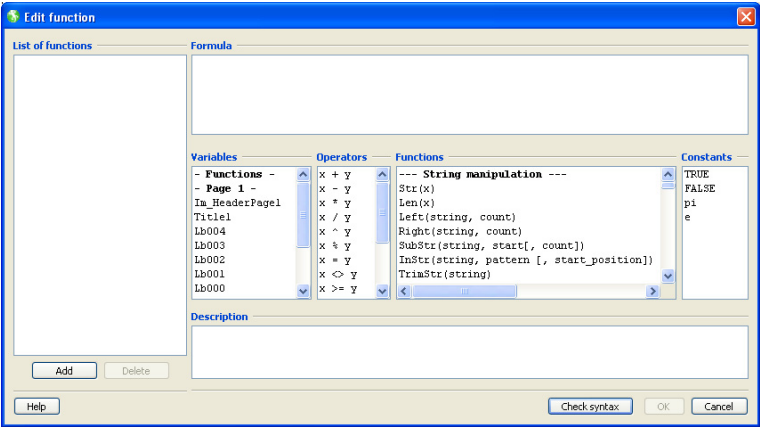
Functions

It is possible that expressions or components of expressions occur in various fields. In this case, it is worthwhile to create functions which can be called by the expressions in the according fields. This considerably reduces the effort to test and maintain the expressions, and errors due to repeated writing the same code become nonexistent.

Functions are built exactly the same way as expressions, and they can be used in the same way. Functions can be considered to be “abbreviations” for components of expressions.

Functions are assembled in the Function editor (see Fig. 4-166).

Fig.4-166 The function editor



The Function editor is identical to use as the expression editor. The main difference is that there is an additional window **Function list** where the function names are listed.

Clicking on **Add** adds a function name to the list. As this default function name is not very self-explanatory, it should be changed to something more suitable for its use. Then the function can be edited when it is selected in the **List of functions**.

Clicking on **Delete** removes the selected function from the list. Note that a deleted function is no longer available, and it is possible that expressions in the form may no longer work properly, because they reference the deleted function.

## 4.5.2

## Actions and logic available in Snapform

The available functions are grouped according to their purpose. The following sections represent the grouping in the expression editor (Chapter 5, "Expression language reference" lists the functions in alphabetical order).

### 4.5.2.1

### String functions

String functions are applied to character strings. Every element can be converted to a string, so that these functions have a wide range of use. The following functions are part of the group of string functions:

#### **Str(x)**

Converts the entered value to a string.

Example see *func\_string1.qdf*

#### **Len(x)**

Returns the length (number of characters) of the string.

Example see *func\_string1.qdf*

#### **Left(string, count)**

Returns the first count characters, counted from left, of the string string.

Example see *func\_string1.qdf*

<b>Right(string, count)</b>	Returns the first count characters, counted from the right, of the string string.  Example see <i>func_string1.qdf</i>
<b>SubStr(string, start[, count])</b>	Creates a substring, beginning at character position start and with length count.  Example see <i>func_string1.qdf</i>
<b>InStr(string, pattern [, start_position])</b>	Returns the position of pattern in the string string, beginning with the search from position start_position.  Example see <i>func_string2.qdf</i>
<b>TrimStr(string)</b>	Removes preceding and trailing spaces from a string.  Example see <i>func_string2.qdf</i>
<b>SubArray(byte_array, start [, count])</b>	Creates a sub-sequence of the byte sequence byte_array, beginning with byte number start and the number of count bytes.  Example see <i>func_string2.qdf</i>
<b>getBytes(string)</b>	Encodes the entered character sequence to a byte sequence which can be passed to a 2D barcode using the encoding Binary.  Example see <i>func_string2.qdf</i>

#### 4.5.2.2

#### Regular Expression functions

Regular Expression functions are an extension of string functions and are used to analyze strings against a pattern defined as a Regular Expression. The following functions are part of the group of Regular Expression functions:

**find(string, regex[, start] )**

Finds the position of the first occurrence of a match after position start of the pattern defined in regex within the entered value string.

Example see *func\_regex.qdf*

**match(string, regex)**

Determines whether the pattern defined in regex is contained in the entered value string.

Example see *func\_regex.qdf*

**replaceAll(orig\_string, regex, replace\_with)**

Replaces all matches of the pattern defined in regex in the entered value orig\_string with the string replace\_with.

Example see *func\_regex.qdf*

#### 4.5.2.3

#### Math functions

Math functions are applied to numbers and run mathematical operations. The following functions are part of the group of math functions:

**Abs(x)**

Calculates the absolute value of the argument.

Example see *func\_math1.qdf*

**Max(x, y)**

Determines the greater of the two values.

Example see *func\_math1.qdf*

**Min(x, y)**

Determines the smaller of the two values.

Example see *func\_math1.qdf*

**Sin(x)**

Calculates the Sine of an angle defined in degrees.

Example see *func\_math2.qdf*

<b>Cos(x)</b>	Calculates the Cosine of an angle defined in degrees.  Example see <i>func_math2.qdf</i>
<b>Tan(x)</b>	Calculates the Tangent of an angle defined in degrees.  Example see <i>func_math2.qdf</i>
<b>Asin(x)</b>	Returns the Arcsine of the argument in degrees.  Example see <i>func_math2.qdf</i>
<b>Acos(x)</b>	Returns the Arccosine of the argument in degrees.  Example see <i>func_math2.qdf</i>
<b>Atan(x)</b>	Returns the Arctangent of the argument in degrees.  Example see <i>func_math2.qdf</i>
<b>Round(x)</b>	Rounds the entered value up or down to the next whole number.  Example see <i>func_math1.qdf</i>
<b>Ceil(x)</b>	Rounds the entered value to the next greater integer.  Example see <i>func_math1.qdf</i>
<b>Floor(x)</b>	Rounds the entered value to the next smaller integer.  Example see <i>func_math1.qdf</i>
<b>Trunc(x)</b>	Rounds the absolute value of the entered value to the next smaller whole number.  Example see <i>func_math1.qdf</i>

<b>Cover(x)</b>	<p>Rounds the absolute value of the entered value to the next greater integer.</p> <p>Example see <i>func_math1.qdf</i></p>
<b>Exp(x)</b>	<p>Calculates the value of the exponential function <math>e^x</math>.</p> <p>Example see <i>func_math2.qdf</i></p>
<b>Log(x)</b>	<p>Calculates the Natural Logarithm of x (<math>\ln x</math>).</p> <p>Example see <i>func_math2.qdf</i></p>
<b>Sqrt(x)</b>	<p>Calculates the square root of the entered value.</p> <p>Example see <i>func_math2.qdf</i></p>
<b>IncVersion(initial_value, step)</b>	<p>Loop counter: sets a counter to initial_value and increments it with each subsequent call by step.</p> <p>Example see <i>func_math1.qdf</i></p>

#### 4.5.2.4

#### Table functions

Table functions are applied on tables and are mainly used for accessing the table data. The following functions are part of the group of table functions:

<b>Total(table, column)</b>	<p>Sums up the values of the column column.</p> <p>Example see <i>func_table1.qdf</i></p>
<b>StrTotal(table, column)</b>	<p>Assembles the table values of column column to a string.</p> <p>Example see <i>func_table1.qdf</i></p>

**TimeTotal(table,  
column)**

Adds the values of the column column to a time.

Example see *func\_table2.qdf*

**Cell(table, column,  
row)**

Represents the value of the table cell of table table, column column, row row.

Example see *func\_table2.qdf*

**TblValues(table,  
tbl\_start, tbl\_end,  
row\_start, row\_end,  
delimiter, column [,  
column[, ...]])**

Creates a string with table values assembled row-by-row and specific delimiters.

Example see *func\_table1.qdf*

**getValues(delimiter,  
value1 [, value2[, ...]])**

Assembles the values of the indicated fields to a string, using the delimiter character delimiter.

Example see *func\_table2.qdf*

**getRow()**

Shows the row number of the according field in the respective table.

Example see *func\_table2.qdf*

**getCol()**

Shows the column number of the according field in the respective table.

Example see *func\_table2.qdf*

### 4.5.2.5

### Date functions

Date functions are applied to date and time, as well as date/time differences.

The following functions are part of the group of date functions:

**Today()**

Returns the current date.

Example see *func\_date.qdf*

<b>Now()</b>	Returns the current time.  Example see <i>func_time.qdf</i>
<b>Date(year, month, day)</b>	Creates a date object from the arguments.  Example see <i>func_date.qdf</i>
<b>Time(hour, minute[, second])</b>	Creates a time object from the arguments.  Example see <i>func_time.qdf</i>
<b>getDay(date_value)</b>	Shows the day of month of the date date_value.  Example see <i>func_date.qdf</i>
<b>getMonth(date_value)</b>	Shows the number of the month of the date date_value.  Example see <i>func_date.qdf</i>
<b>getYear(date_value)</b>	Shows the number of the year of the date date_value.  Example see <i>func_date.qdf</i>
<b>getHour(time_value)</b>	Shows the hour number of the time time_value.  Example see <i>func_time.qdf</i>
<b>getMinute(time_value)</b>	Shows the minute number of the time time_value.  Example see <i>func_time.qdf</i>
<b>addDay(date_value, number)</b>	Calculates the date which is from the date date_value away by number of days.  Example see <i>func_date.qdf</i>



**addMonth(date\_value, number)** Calculates the date which is from the date date\_value away by number months.

Example see *func\_date.qdf*

**addYear(date\_value, number)** Calculates the date which is from the date date\_value away by number years.

Example see *func\_date.qdf*

**addHour(time\_value, number)** Calculates the time which is from the time time\_value apart by number of hours.

Example see *func\_time.qdf*

**addMinute(time\_value, number)** Calculates the time which is from the time time\_value apart by number of minutes.

Example see *func\_time.qdf*

**calcDays(date\_first, date\_second)** Calculates the number of days between the entered dates.

Example see *func\_date.qdf*

**calcHours(date\_first, date\_second)** Calculates the number of hours between the entered dates.

Example see *func\_time.qdf*

**calcMinutes(date\_first, date\_second)** Calculates the number of minutes between the entered dates.

Example see *func\_time.qdf*

#### 4.5.2.6

#### State functions

State functions are used to set a state (printable, visible, writable) of fields, pages and categories. The following functions are part of the group of state functions:

<b>setVisible(category_list, true false)</b>	Sets the visible property of the affected category(s). Example see <i>func_status.qdf</i>
<b>setPrintable(category_list, true false)</b>	Sets the printable property of the affected category(s). Example see <i>func_status.qdf</i>
<b>setWritable(category_list, true false)</b>	Controls the read-only property of the according category(s). Example see <i>func_status.qdf</i>
<b>setFieldVisible(true false, field_name)</b>	Sets the visible property of the field. Example see <i>func_status.qdf</i>
<b>setFieldPrintable(true false, field_name)</b>	Sets the printable property of the field. Example see <i>func_status.qdf</i>
<b>setFieldWritable(true false, field_name)</b>	Controls the read-only property of the field. Example see <i>func_status.qdf</i>
<b>setPageVisible(true false, page_number [, page_number[, ...]])</b>	Sets the visible property of the affected page(s). Example see <i>func_status.qdf</i>
<b>setPagePrintable(true false, page_number [, page_number[, ...]])</b>	Sets the printable property of the affected page(s). Example see <i>func_status.qdf</i>

#### 4.5.2.7

### Consistency check functions

Consistency check functions are used to create and verify checksums over entered data. The following functions are part of the group of consistency check functions:

<b>checkCardNumber(value)</b>	Validates credit card numbers.  Example see <i>func_consist.qdf</i>
<b>getMod10(value)</b>	Returns a checksum created with the "MOD-10 algorithm" over the entered value value.  Example see <i>func_consist.qdf</i>
<b>checkMod10(value)</b>	Validates number using the "MOD-10 algorithm".  Example see <i>func_consist.qdf</i>
<b>getMod10r(value)</b>	Returns a checksum created with the "Recursive MOD-10 algorithm" over the entered value value.  Example see <i>func_consist.qdf</i>
<b>checkMod10r(value)</b>	Validates number using the "Recursive MOD-10 algorithm".  Example see <i>func_consist.qdf</i>

#### 4.5.2.8

#### Identifier functions

Identifier functions are used to create unique strings which allow an exact association with data and records. The following functions are part of the group of identifier functions:

<b>createGUID()</b>	Creates an unique identifier (GUID, Global Unique Identifier).  Example see <i>func_ident.qdf</i>
<b>getSessionID()</b>	Creates an unique identification string (GUID, Global Unique Identifier), which can be assigned to the current working session with Snapform Viewer.  Example see <i>func_ident.qdf</i>

**getPrintID()** Creates an unique identification string (GUID, Global Unique Identifier), which can be linked to a print job.

Example see *func\_ident.qdf*

**getDataID()** Creates an unique identification string (GUID, Global Unique Identifier), which can be linked to a set of forms data.

Example see *func\_ident.qdf*

**getRandomString(length [, digits])** Creates a randomly assembled string of the length length.

Example see *func\_ident.qdf*

**getPrintIdStr(length [, digits])** Creates a freely configurable identifier string which can be linked to a print job.

Example see *func\_ident.qdf*

#### 4.5.2.9

#### Combo box functions

Combo box functions are used to manage and configure combo box form elements. The following functions are part of the group of combo box functions:

**setList(ComboBox, str\_items, str\_values)** Configures the selection list for the list field ComboBox.

Example see *func\_combobox.qdf*

**getSelectedIndex(ComboBox)** Returns the index number of the current selection of a List field.

Example see *func\_combobox.qdf*

**setSelectedIndex(ComboBox, index)** Sets the current selection of the List field ComboBox to the value associated to index index.

Example see *func\_combobox.qdf*

#### 4.5.2.10

### Special functions

Special functions are functions which cannot be assigned to any of the other groups. The following functions are part of the group of special functions:

#### **Compress(x)**

Creates a ZLIB compressed byte sequence from the entered value, which may be used in a barcode using the encoding method Binary.

Example see *func\_barcodes.qdf*

#### **if(logical\_test, value\_if\_true [,value\_if\_false])**

Executes a comparison operation on the specified operands and returns a value depending on its result.

Example see *func\_misc.qdf*

#### **isEmpty(x)**

Tests whether the argument x is empty (or null).

Example see *func\_misc.qdf*

#### **importData(url)**

Imports a data file from url.

Example see *func\_misc.qdf*

#### **getVersion()**

Returns the version number of the running Snapform application (Designer, Viewer).

Example see *func\_ident.qdf*

#### **getContent(url)**

Loads the (text) contents of url.

Example see *func\_misc.qdf*

## 4.6 Integrity of the form

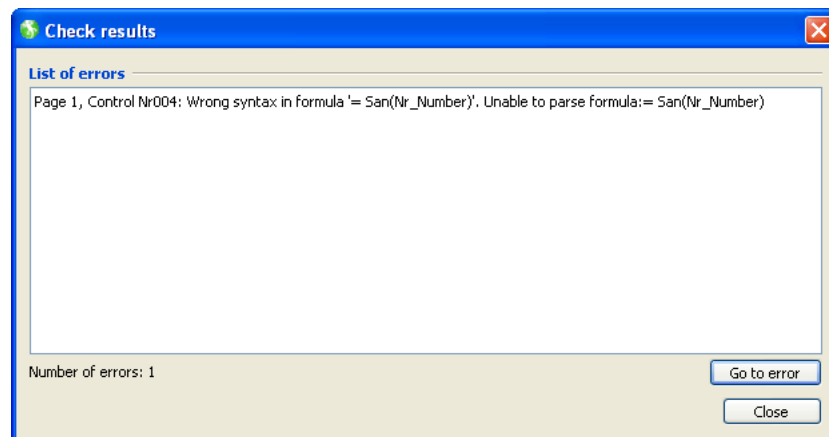
One of the most important steps when finishing a form is to ensure its integrity. That means that the form is in complete working order.

The first step is checking the form to find possible internal errors. This is accomplished with the **Check form** function (Toolbar or **Tools** menu).

Since it is possible to save a form and continue working with it, even if there are erroneous expressions, this test is important, because it reveals such errors.

When errors are found, the error list opens (see Fig. 4-167).

**Fig.4-167** Error list from  
**Check form**



In this example, an error was found, on page 1 in the number field **Nr004**. The expression is wrong; the function is not named **San()**, but **sin()**. The line is selected, and after clicking on **Go to error**, the affected form element becomes active, and the error can be fixed.

When this function doesn't show any more errors, it means that the form works. That doesn't mean the form works correctly, because conceptional and logical errors cannot be found with this function.

The next step in checking the form is testing with data. Test it using "possible" data entry values, in order to verify the results. Testing with "impossible" values and "intentionally wrong" values is just as important.

## 4.7

## Stylesheets

Stylesheets control the appearance of form elements. Among the features controlled are font type and font size, background color, and border appearance depending of the context of the functionality, as well as the appearance of tables.

Stylesheets can be exported and imported, so it is possible to create forms with a consistent appearance.

The stylesheets are stored within their form. Every form carries its own stylesheets.

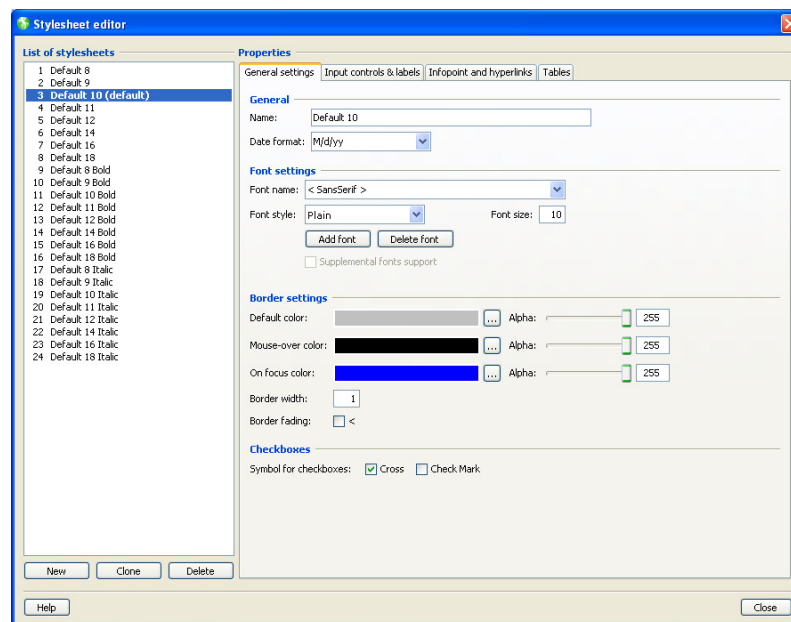
The stylesheets are edited in the stylesheet editor.

### 4.7.1

### The Stylesheet editor

The Stylesheet editor (see Fig. 4-168) is started with the **Stylesheet editor** in the **Tools** menu.

**Fig.4-168** *The Stylesheet editor, General settings tab*



The Stylesheet editor provides four settings tabs for a completely specified stylesheet. The **General settings** contain fonts, base date format and general border settings for fields.

In the List of stylesheets on the left side, all stylesheets in the current document are listed. The stylesheets with **Default** in their names are default stylesheets and are part of any newly created form. In earlier versions of Snapform, the stylesheets with the font names **Arial** and **Courier** were the default stylesheets.

Clicking on the **New** button adds a new stylesheet. Clicking on the **Clone** button creates a copy of the currently selected stylesheet and adds it to the list. Clicking on the **Delete** button removes the current (highlighted) stylesheet.

The field **Name** holds the name of the stylesheet. This is the name under which the stylesheet is listed in the **Stylesheet** property in the form.

The **Date format** field displays a drop-down list of default date formats. This default format can be overridden in the **Date format** property of the field. The format specified in the stylesheet is also used for entering dates into the field.

In the **Font settings** area, the font used in the form element is selected. The drop-down menu **Font name** is a list of available fonts. These font names are either actual fonts installed in the system (such as Arial or Times), or logical fonts (such as **SansSerif** or **Serif**). Logical fonts are fonts where just the main feature is specified, but the font actually used is defined on operating system level.

By clicking on the **Add font** button, additional fonts installed in the system can be added for use in Snapform. A file open dialog opens where the font files can be selected. At the moment, Snapform supports only the TrueType font format (.ttf).

**Note:** *Fonts are available with a wide variety of licensing models. It is fully within the responsibility of the user of Snapform to make sure that these license terms are accordingly honored.*

Clicking on the **Delete font** button removes the selected font from the list.



Snapform has the option of supplemental font support, which allows you to specify the exact font depending on the platform. When the **Supplemental font support** check box is active, the Supplemental font support area is shown, which consists of three entry fields **Font name Windows**, **Font name Mac OS X** and **Font name Unix**, where the according font can be specified.

The options from the **Border settings** area are valid for all form elements (as far as borders are activated in the respective properties).

The **Default color** is the color the border has in its normal appearance. Clicking on the ... button opens the color picking window (see section 3.2.9.2), and the appropriate color can be selected. The **Alpha** slider controls the transparency of the color. The lower this value is, the more transparent the border (or the area) of the element is rendered. The value for **Alpha** is displayed in a field right of the slider, and it can be entered directly. The value can be between **0** and **255**.

**Note:** *The principles for setting the color are valid for all areas of the stylesheet editor.*

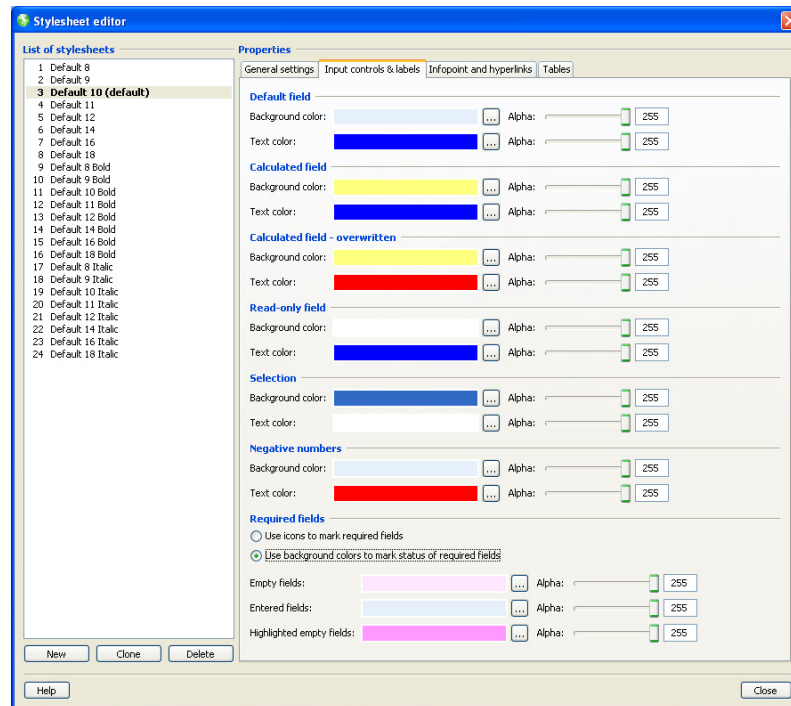
**Mouse over color** is the color of the border when the mouse cursor is within the perimeter of the element.

**On focus color** is the color of the border when the element has the focus (is active for entering information). This is, for example, the case when the field has been tabbed into.

**Border width** sets the width of the border in Points. Default value is **1**.

When the check box **Border fading** is active, the border color fades in and out when mouse enters and exits the field.

**Fig. 4-169** *The Stylesheet editor, Input controls and labels tab*



In the **Input controls and labels** settings, the background and text color for various states of input fields is controlled. **Background color** is the color of the background of the field, if it has not been overridden in the Back color property of the element. **Text color** is the color of the text, if it has not been overridden in the properties of the element.

The **Default field** setting is, as the name says it, the default setting for the (empty) element.

**Calculated field** contains the settings for a field which displays the result of a calculation.

**Calculated field - overwritten** contains the settings for a field which displays the result of a calculation, but has been manually overwritten.

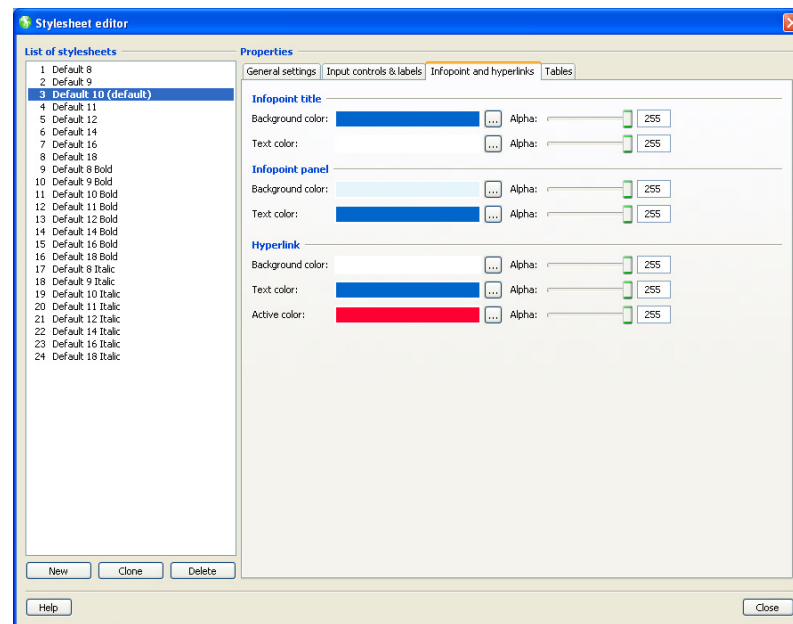
**Read-only field** contains the settings for a field which has been made non-writable by setting the **read-only** property.

**Selection** contains the settings for a list box or combo box field.

**Negative numbers** contains the settings when the value of a number field is negative.

**Required fields** controls the display behavior of fields which have been marked as required. When the option **Use icon to mark required fields** has been selected, required fields will display an icon representing an exclamation mark. When the option **Use background colors to mark status of required fields** has been selected, three additional color selectors appear. **Empty fields** sets the color of the field when it is empty. **Entered field** shows the color when the field has been filled, and **Highlighted empty fields** shows the color for an empty field which has the focus.

**Fig.4-170** *The Stylesheet editor, Infopoints and hyperlinks tab*



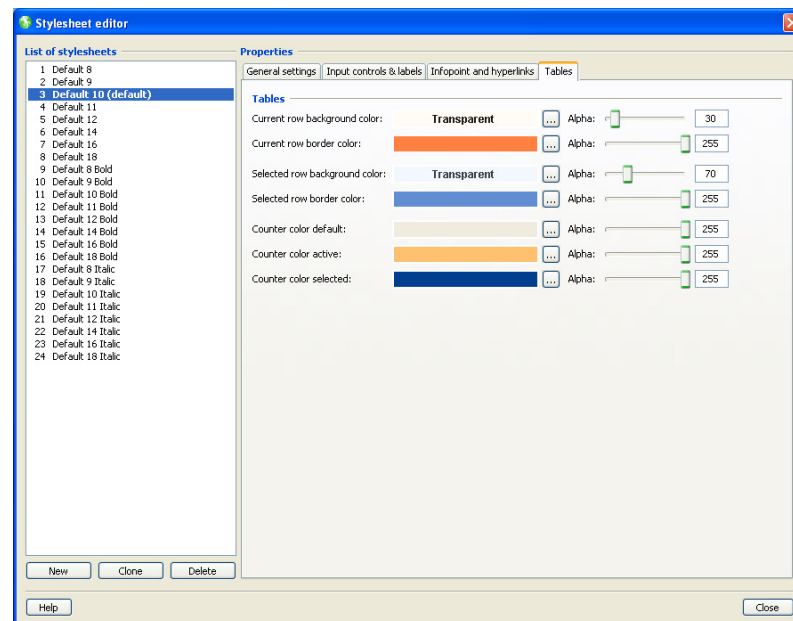
In the **Infopoints and hyperlinks** settings, the background and text color for various states of infopoints and hyperlinks is controlled. **Background color** is the color of the background of the element, if it has not been overridden in the Back color property. **Text color** is the color of the text, if it has not been overridden in the properties of the element.

**Infopoint title** contains the settings for the title bar of opened infopoints.

**Infopoint panel** contains the settings for the display area of infopoints.

**Hyperlink** contains the settings for hyperlinks. Besides the setting for background color and text color, the setting for the Active color can be specified, which is applied when the link is active (clicked on).

**Fig.4-171** *The Stylesheet editor, Tables tab*



In the **Tables** settings, the color for the background of the row and the selectors is controlled.

**Current row background color** is the highlight color for the current row (the row which contains a field which is active). **Current row border color** is the color of the border of the highlighted area of the current row.

**Selected row background color** is the highlight color for the selected row (selected by clicking on the selector field). **Selected row border color** is the color of the border of the selection area of the selected row.

**Counter color default** is the color of the selector when it is in its normal state (neither selected nor active). **Counter color active** is the highlight color for the selector of the active row (a field in the row is active).

**Counter color selected** is the color of the border of the selected row which has been selected by clicking on the selector.

## 4.7.2

### Importing and exporting stylesheets

Stylesheets are imported and exported with the **Import stylesheet** and **Export stylesheet** menu items. When selected, an Open file (importing) or Save file (exporting) dialog displays, allowing you to specify the file name.

## 4.8 Form and security settings

### 4.8.1 Introduction

The last step before the form can be finalized and published is the configuration of the form and security settings. These settings concern various aspects for using the form (such as the configuration of the action buttons), but also the premises for embedding into workflows and preparations for archiving.

These settings are done in the **Form settings** window. This window opens with the menu item **Tools —> Form settings**.

The Form settings consists of 7 dialogs which are selected from the selection list at the left border of the window. These dialogs are explained in the following.

As the default settings in these dialogs are essentially empty fields, this configuration must be completed for each form.

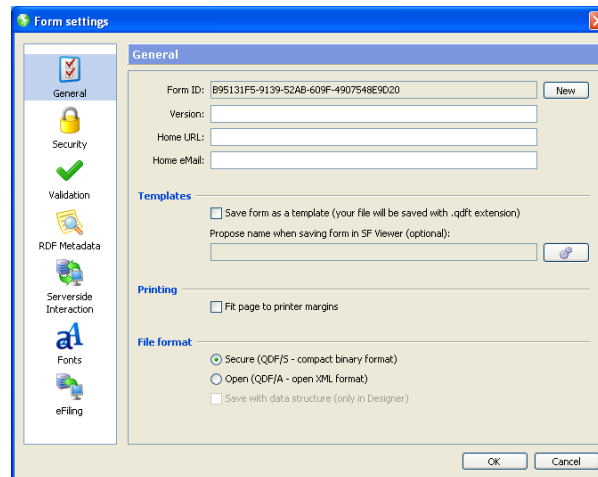
**Note:** *The results of the configuration settings must be verified in the Snapform Viewer, because the Snapform Designer cannot display certain functions.*

**Note:** *Some settings must also be used in conjunction with server configurations. References to server-side applications, their functions and configurations are beyond the scope of this manual. For these functionalities we refer you to the manuals for those systems.*

### 4.8.2 General

General settings (see Fig. 4-172) is the dialog which is displayed when the Form settings window opens. The window is subdivided into different zones: **General**, **Templates**, **Printing** and **File format**.

**Fig. 4-172** *The Form settings:  
General window*



### 4.8.2.1

## Document information

The document information is about the document itself, and its components can, for example, be used to identify it in forms or document management systems.

## Form-ID

The Form-ID is a unique identifier in the GUID format which provides a definite identification of the document worldwide. This value is created with the creation of the document, and remains saved with it.

Clicking on **New** allows you to create another Form-ID. A dialog asks whether the ID should be changed, and when confirmed, the change is executed.

## Version

In this text field, the version of the document can be entered. This value appears in the metadata of the file information.

## Home URL

URL of the issuer of the form. This URL may be used for providing updates to the form, or to allow contact with the issuer or its technical support.

## Home eMail

E-Mail address of the issuer of the form. This address can, for example, be used for contacting the issuer, or their support department.

### 4.8.2.2

## Templates


The form can be saved as a template. This always resets the form when it gets opened, and it cannot be saved under the same name. This is very useful in places where a centralized form repository is in use.

### Save Form as a template

When this box is checked, the form is saved as a Template in the **.qdft** format.

### Propose name when saving form in SF Viewer

The name under which the (filled out) form is saved from Snapform Viewer can be suggested via this field. Such a suggestion is particularly important if the form is part of a document-based workflow, and the file name has a meaning. The suggestion may either be static (a fixed text) or dynamic, as the result of an expression. The structure of the suggested name depends on the actual use of the form.

Clicking on the  icon opens the expression editor. Here, the expression to calculate the suggested file name can be entered. Part of that suggestion could be, for example, the current date, first name and last name, or any other unique identification. The expression editor has access to all available fields and functions.

### 4.8.2.3

## Printing

### Fit page to printer margins

This check box controls whether the form is scaled when it is printed, that it fits completely within the printable area of the selected printer. This ensures the form will be completely printed.

If scanners (with optical character recognition) are used in the post-processing of the printed forms, it must be verified whether scaling is acceptable. If there are doubts, it is recommended to uncheck the **Fit page to printer margins** check box, and instead provide sufficiently wide margins in the form, so it can be properly printed.

This box must be unchecked when the form is printed of preprinted stock.



#### 4.8.2.4

### File format

The file format can be selected when a new document is created. It is, however, not known at that moment whether the data will be saved encrypted or in the clear. This is why the file format can be specified in the Form settings.

### Secure / Open

These radio buttons specify whether the data will be encrypted and compressed, or left in the clear in XML form when the form is saved.

The **Secure** option saves the form in the QDF/S format, which is very compact. In order to retrieve the data, a special server component is required (further information about such a component is available from Ringler Informatik AG).

The **Open** option saves the form in the QDF/A format which is less compact, but makes the data accessible to any XML parser (or other application).

The **Save with data structure** check box specifies whether the complete empty data structure is created when the form is saved in the Snapform Designer. This is necessary when the form is used as a template and pre-filled on a server. This option also makes work easier when developing the connection to a back-end process.

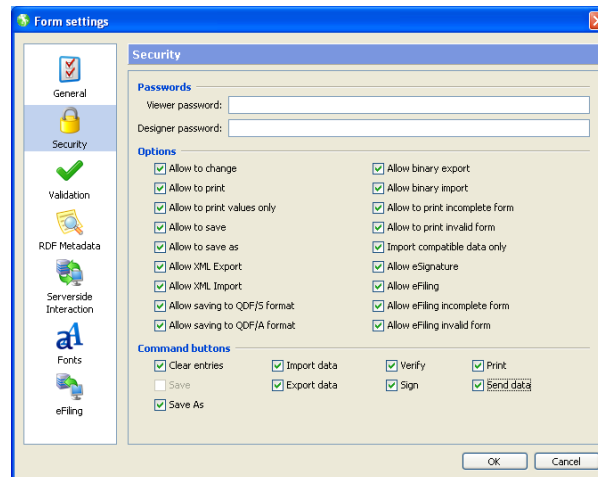
**Note:** *This option is limited to the Snapform Designer. When saving the form in the Snapform Viewer, only the fields with an actual value are passed to the data structure.*

#### 4.8.3

### Security

In the **Security** dialog (see Fig. 4-173) the document settings which relate to security or permissions of the form are displayed. This includes access to the document and the usage rights.

**Fig. 4-173** *The Form settings:  
Security window*



The Security dialog is subdivided into three areas: **Passwords**, **Options** and **Command buttons**.

#### 4.8.3.1

#### Passwords

The document can be protected with two types of passwords. The entry of these values into the corresponding field occurs in the clear, and the passwords will be displayed in the clear after saving and reopening the document in the Snapform Designer.

The protection using passwords is effective to a certain degree. If a more extensive access protection to the form is needed, external means must be used. Ringler Informatik AG provides support for the implementation of extended security mechanisms.

#### Viewer password

This password is requested when opening the form for regular use. It allows you to control the access to the form itself. Without providing the correct Viewer password, the form cannot be opened in the Snapform Viewer.

#### Designer password

This password is requested when opening the form in the Snapform Designer. This allows you to control the access to the form, and in

particular to its business logic. Without providing the correct Designer password, the form cannot be opened in the Snapform Designer.

#### 4.8.3.2

### Options

In the Options area, the various usage rights for the form are listed, and they can be activated or deactivated individually. These options are fully honored in the Snapform Viewer.

#### **Allow to change**

When this box is checked, the form can be changed in the Snapform Designer.

#### **Allow to print**

When this box is checked, the form can be printed.

#### **Allow to print values only**

When this box is checked, only the values from the entry fields can be printed (this option is, for example, necessary when the printing occurs on pre-printed stock).

#### **Allow to save**

When this box is checked, the form can be saved using the Save command. It is, however, not possible to save the form to another location or another name using the Save as... command.

#### **Allow to save as**

When this box is checked, the form can be saved using the Save as... command. It is possible to save to another location and another name.

#### **Allow XML export**

When this box is checked, the form data can be exported in XML.

#### **Allow XML import**

When this box is checked, the form data can be imported in XML.

#### **Allow saving to QDF/S format**

When this box is checked, the form can be saved in the compact binary format (QDF/S).

#### **Allow saving to QDF/A format**

When this box is checked, the form can be saved in the open accessible format (QDF/A).

### Allow binary export

When this box is checked, the form data can be exported in the encrypted binary data format.

### Allow binary import

When this form is checked, form data in the encrypted binary data format can be imported.

### Allow to print incomplete form

When this box is checked, the form can be printed, even if it is not completely filled out. The completeness is controlled via a rule specified in the **Validation** dialog (see section 4.8.4.3).

### Allow to print invalid form

When this box is checked, the form can be printed, even if it is not filled out with valid information. The validity is controlled via a rule specified in the **Checking** dialog (see section 4.8.4.2).

### Import compatible data only

When this box is checked, data can only be imported when it is compatible with the form.

### allow eSignature

When this box is checked, the form can be digitally signed with appropriate systems.

### Allow eFiling

When this box is checked, the form can be filed electronically using the **eFiling** mechanism (defined in the eFiling dialog, see section 4.8.8).

The next two options are only active when this box is checked.

### Allow eFiling incomplete form

When this box is checked, the form can be filed electronically even if it is not completely filled out. The completeness is defined in the rule defined in the **Validation** dialog (see section 4.8.4.3). This check box is only available when the **Allow eFiling** box is checked.

### Allow eFiling invalid form

When this box is checked, the form can be filed electronically even if it is not filled out with valid information. The validity is defined in the rule defined in the **Validation** dialog (see section 4.8.4.2). This check box is only available when the **Allow eFiling** box is checked.

### 4.8.3.3

## Action keys

One of the features of Snapform is that action buttons are moved out of the form to the application. In most form systems, the essential actions (Reset, Print, Save, Submit) must be individually designed and configured for each and every form. In Snapform, the form holds the information concerning how the functionality should be supported, and the Snapform Viewer provides it.

In the Command buttons area, the functionality the form shall support is selected. In order to use the functionality, the form may need the according right (checking the according box in the **Options** area (see section 4.8.3.2)).

### Clear entries

When this box is checked, the Snapform Viewer displays the **Clear entries** button. When this button is clicked in the Snapform Viewer, the form is reset.

### Save

When this box is checked, the Snapform Viewer displays the **Save** button. When this box is checked, the **Save as** check box becomes inactive. When this button is clicked in the Snapform Viewer, the form is saved under its same name.

### Save as

When this box is checked, the Snapform Viewer displays the **Save as** button. When this box is checked, the **Save** check box becomes inactive. When this button is clicked in the Snapform Viewer, the form is saved under a different name (which may be proposed by the form, or freely specified by the user).

### Import data

When this box is checked, the Snapform Viewer displays the **Import data** button. When this button is clicked in the Snapform Viewer, data will be imported into the form.

## Export data

When this box is checked, the Snapform Viewer displays the **Export data** button. When this button is clicked in the Snapform Viewer, data will be exported from the form.

## Verify

When this box is checked, the Snapform Viewer displays the **Verify** button. When this button is clicked in the Snapform Viewer, the form will be validated according to the specified validation rules.

## Sign

When this box is checked, the Snapform Viewer displays the **Sign** button. When this button is clicked in the Snapform Viewer, the process to digitally sign the form is initiated.

## Print

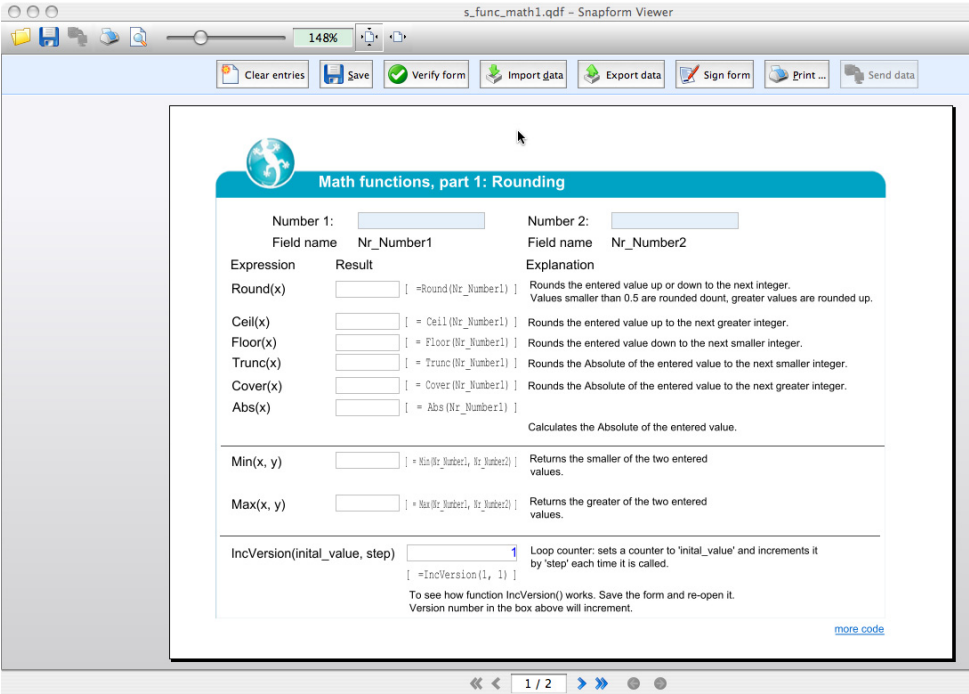
When this box is checked, the Snapform Viewer displays the **Print** button. When this button is clicked in the Snapform Viewer, the form is printed to the default printer.

## Send data

When this box is checked, the Snapform Viewer displays the **Send data** button. When this button is clicked in the Snapform Viewer, the form data will be submitted to the specified server (eFiling).

When all check boxes are selected, the form appears in the Snapform Viewer as shown in Fig. 4-174.

**Fig.4-174** *The Actions toolbar in the Snapform Viewer*



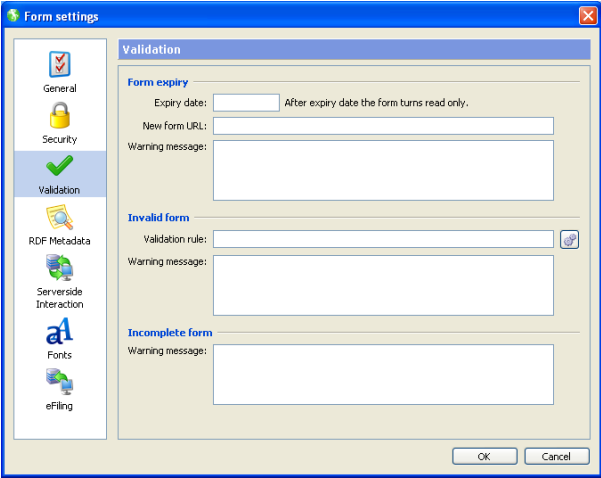
Below the regular toolbar, a new toolbar is displayed which contains the Actions which have been activated in the form settings.

4.8.4

Validation

The **Validation** dialog (see Fig. 4-175) contains document settings which concern the validity of the form or its data.

**Fig.4-175** *The Form settings: Validation window*



The **Validation** dialog is subdivided into three areas: **Form expiry**, **Invalid form** and **Incomplete Form**.

#### 4.8.4.1

### Form expiry

It is possible to add an expiration date to the form. This is particularly useful when the data contained in the form is relevant for a specific time frame, and a different form must be used for later dates.

#### Expiry date

In this field, the expiration date is entered. The date is compared with the system clock of the host computer, and when the current date is after the expiration date, the form will be opened in read-only mode, and no more entries can be made.

If a “stronger” expiration date control is required (rather than using the system clock), an external solution must be used. Ringler Informatik AG provides support for the implementation of such solutions.

#### New form URL

When an expiration date is specified, it is highly recommended to provide a reference to the currently valid form. This reference may be a URL which is specified in this field.

#### Warning message

When an expired form is opened, the user must be notified. The text for this notification is application-dependent, and can therefore not be specified in the application. This field allows you to specify the text to be displayed when the form is opened after it has expired.

#### 4.8.4.2


### Invalid form

In order to prevent submitting invalid data, which can be an issue with electronic forms (and might even lead to a lock-down of the back end system), the form's data can be validated. This validation is done according to a validation rule which is specified in this zone of the dialog.



## Validation rule

The validation rule is an expression which must return either **TRUE** or **FALSE**. When the returned value is **TRUE**, the form is valid, otherwise, it is not valid.

Clicking on the Expression icon  opens the expression editor to enter the expression. All field objects and functions are available in the expression editor.

## Warning message

When the form is considered to be invalid, the user must be notified. This is done with the freely specifiable message from this field. The text may be displayed in the form of a system message.

### 4.8.4.3

## Incomplete form

Besides to the validity of a form, you can also check the form's completeness, by testing the form to see if all fields where the Required field property is checked have an entry. This means that there is no need for an explicit validation rule.

## Warning message

When the form is considered to be incomplete, the user must be notified. This is done with the freely specifiable message from this field. The text may be displayed in the form of a system message.

### 4.8.5

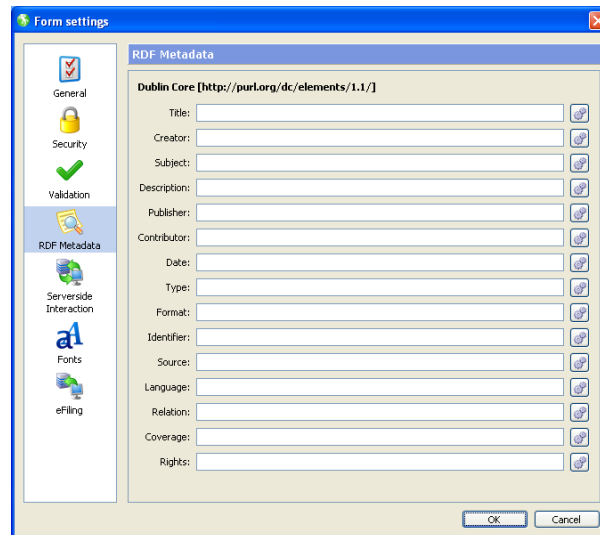
## RDF Metadata

For managing and archiving of documents, properly established metadata is a fundamental issue. Programmatic creation of the metadata ensures that additional sources of error by subsequent manual entry are avoided.


Snapform allows you to create elements of the *Dublin Core Metadata Initiative Definition 1.1* to be specified static, or as the result of an expression.

The **RDF Metadata** dialog (see Fig. 4-176) lists the metadata elements which are relevant for forms.

**Fig. 4-176** *The Form settings:  
RDF Metadata  
window*



The **RDF Metadata** dialog consists of entry fields for the various metadata elements. For an explanation of the meaning of these elements, we refer to the documentation of the Dublin Core Metadata Initiative (<http://www.dublincore.org>).

The values can be entered either as static text, or as the result of an expression. For every element, clicking on the Expression icon  opens the expression editor to enter the expression. All field objects and functions are available in the expression editor.

## 4.8.6

## Server-side interaction

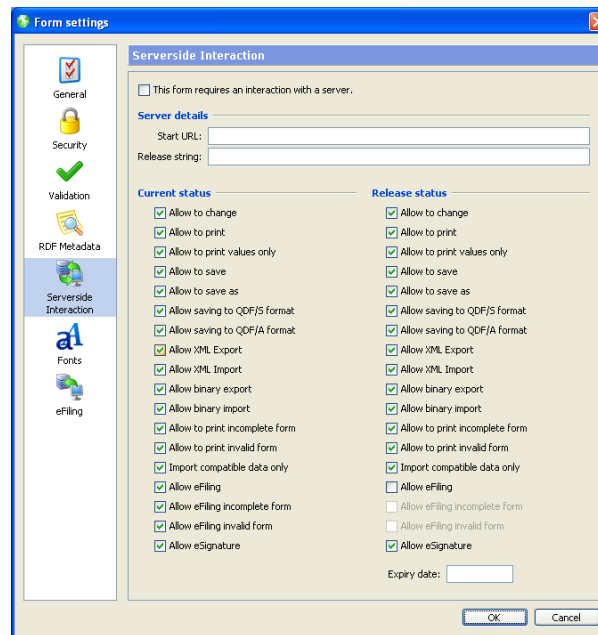
Snapform provides the possibility to change the rights of a form (see section 4.8.3.2) using a server-side command. This feature is called “release”. Possible applications are, for example, to protect the form after it has been eFiled, or releasing additional functions after the clearance from a payment system.

The server-side interaction involves calling a specific URL, which returns a particular string. When this string corresponds to the specified release string, the release action occurs, otherwise nothing happens (unless the result is evaluated in an expression, which returns a message to the user).

**Note:** The term “release” means a change of the state of the form which is initiated by an interaction with the server. It does not necessarily mean that the form has more rights than before.

The **Serverside interaction** dialog (see Fig. 4-177) contains the information needed for the communication with the server.

**Fig.4-177** The Form settings:  
Serverside  
interaction  
window



The **Serverside interaction** dialog contains the information for releasing and the settings state for the form before and after the release.

### This form requires an interaction with the server

This check box is the “main switch” for the server-side interaction. When it is checked, the mechanism is active, otherwise all further settings of this dialog are ignored.

The **Server details** zone contains the information for the server request.

### Start URL

This is the URL which initiates the server request for release.

### Release string

This is the string which the server must return to allow the release. When this string is received from the server, the form is released.

In the **Current status** zone, the options already set in the security settings (see section 4.8.3.2) are repeated. They can also be changed in this dialog.

In the **Release status** zone, the options for the released state, after the release, are specified. These options must be set in this dialog.

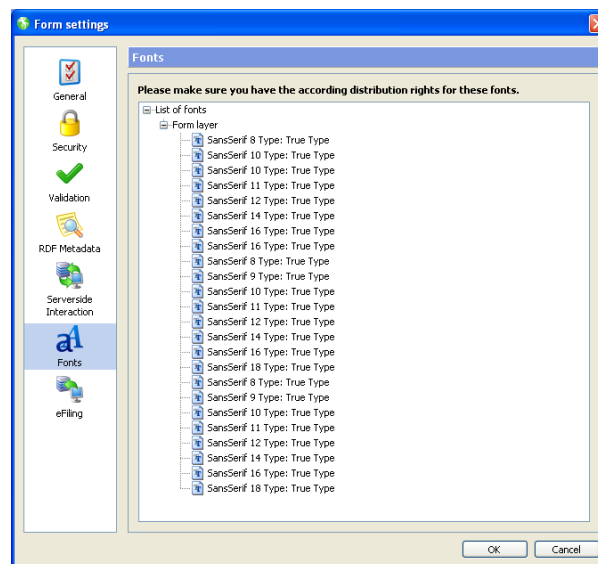
in addition to these options, there is a field **Expiry date**, in which an expiration date for the released state can be specified (see also section 4.8.4.1).

## 4.8.7

## Fonts

The **Fonts** dialog (see Fig. 4-178) shows all the fonts used in the form (or defined in the stylesheets of the form).

**Fig.4-178** The Form settings:  
Fonts window



The **Fonts** dialog is an overview of the fonts referred to in the form, and it is primarily an aid to the form developer to determine whether the license terms for the various fonts are honored.

**Note:** *It is in the sole responsibility of the form developer to ensure that the license terms associated with the various fonts are honored. Ringler Informatik AG refuses any responsibility for possible font license violations.*

This dialog only displays information.

## 4.8.8

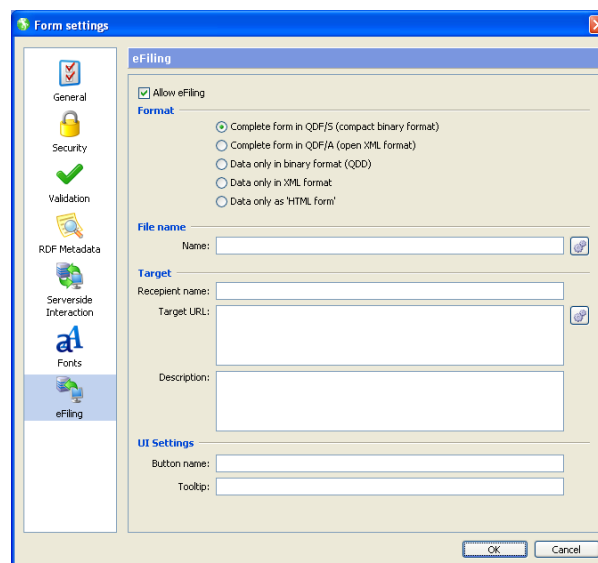
## eFiling

One of the great advantages of electronic forms is the ability to transfer seamlessly the information to the associated business process. This data transfer is called, particularly in government environments, “eFiling”.

It is clear that for a successful eFiling, the form and an according back-end process must be matching. This also means the form developer must cooperate with the respective IT departments.

The **eFiling** dialog (see Fig. 4-179) contains the information relevant for the data transfer and the eFiling.

**Fig.4-179** *The Form settings:  
eFiling window*



The **eFiling** dialog is subdivided into the areas **Format**, **File name**, **Target** and **UI settings**.

## Allow eFiling

This check box is the “main switch” for the eFiling capability. When it is checked, the mechanism is active, otherwise all further settings of this dialog are ignored.

#### 4.8.8.1

### Format

The eFiling allows using several data formats. The preferred format depends on the legal/organizational premises, and on the architecture of the back-end systems.

The selection of the format is made with a set of radio buttons:

#### Complete form in QDF/S (compact binary format)

The form is saved in the compact QDF/S format and submitted as a whole. In order to enable this submission method, the form options **Allow eFiling** and **Allow saving to QDF/S format** must be selected. This eFiling method requires special server tools to retrieve the data .

#### Complete form in QDF/A (open XML format)

The form is saved in the open QDF/A format and submitted as a whole. In order to enable this submission method, the form options **Allow eFiling** and **Allow saving to QDF/A format** must be selected. This eFiling method does not require special server tools to retrieve the data .

#### Data only in binary format (QDD)

The form data is submitted in the binary QDD format and submitted without the base document. In order to enable this submission method, the form options **Allow eFiling** and **Allow binary export** must be selected. This eFiling method requires special tools to retrieve the data on the server.

#### Data only in XML format


The form data is submitted in the XML format and submitted without the base document. In order to enable this submission method, the form options **Allow eFiling** and **Allow XML export** must be selected. This eFiling method does not require special tools to retrieve the data on the server.

#### Data only as 'HTML form'

The form data is submitted as HTML using the POST Method, without the base document. In order to enable this submission method, the form options **Allow eFiling** must be active. With this transmission method, the same scripts and applications can be used on the server side that would have been used with HTML forms.

#### 4.8.8.2

##### File name

When files are transferred, their name must be specified in advance. This may either be a static value (recommended only for special cases) or it may be a dynamically assembled name using an expression. Clicking on the expression icon  opens the expression editor. All field objects and functions are available in the expression editor.

#### 4.8.8.3


##### Target

In this area, the information for the receiving system is specified.

##### Receipient name

This field allows you to specify an additional reference for the post processing units (such as name of the department).

##### Target URL

This is the URL to which the transmission will go. This may either be a static value or the result of a calculation. Clicking on the expression icon  opens the expression editor. All field objects and functions are available in the expression editor.

##### Description

This is the descriptive text for the data transfer.

#### 4.8.8.4

##### UI settings

In this area, the **eFiling** button which appears in the Snapform Viewer, and is controlled in the security settings for the form (see section 4.8.3.3), can be modified to represent closer the meaning of the eFiling process (such as "Pay", "File", "Submit").

##### Button name

In this field, the label of the **eFiling** button in the Snapform Viewer is specified. No entry means that the default label **eFiling** will be used.

##### Tooltip

In this field, the tooltip text of the **eFiling** button in the Snapform Viewer is specified. No entry means that the default text will be used.

## 4.9 Release and deployment

The form has been prepared for distribution in the steps explained in section 4.8. In order to distribute it reliably, it needs to be thoroughly tested. These tests should contain data transfer and database connectivity if these features are used.

If there are a certain number of forms to be distributed, it is recommended that a formal release process be set up. This is a necessity when the forms are in an environment which has advanced requirements concerning the documentation of processes.

The release process is essentially an organizational issue, and cannot be supported any further with the Snapform Designer. Forms or document management tools can supervise and document these processes. For the selection and implementation of such processes, support is provided by Ringler Informatik AG.

The actual release on a technical level is very easy. The forms can be made available in a web environment for the Intranet or the Internet just like any other document. If access must be restricted, the same measures are to be taken as to restrict access to any other web content. Snapform forms are always a file download, which means there are no further configuration changes necessary on the server-side (which is, only true for deploying the forms; for accepting filled out forms and data, adaptations on the server-side are necessary).

When the forms are to be deployed on media, the procedure is straightforward as well. Snapform forms are self-contained files, and can be copied to media or file servers like any other file.

Deployment of in-house created Snapform forms is freely available, and the distribution rights are granted with the Snapform Designer license. Redistributing the Snapform Viewer (for example on CD-ROM) is allowed, but you must make sure the user is encouraged to download the newest version of the Snapform Viewer.



## 4.10 Forms management

Forms management is very important in fields which have to comply to extended requirements concerning security or traceability. Proper version management is a necessity. Version management is supported in Snapform with the Form-ID in the General form settings (see section 4.8.2) as well as the metadata (see section 4.8.5).

It is highly recommended that you set up a forms management program which complies to the requirements of your organization. For the specification and implementation of a forms management system, Ringler Informatik AG provides support and consulting services.

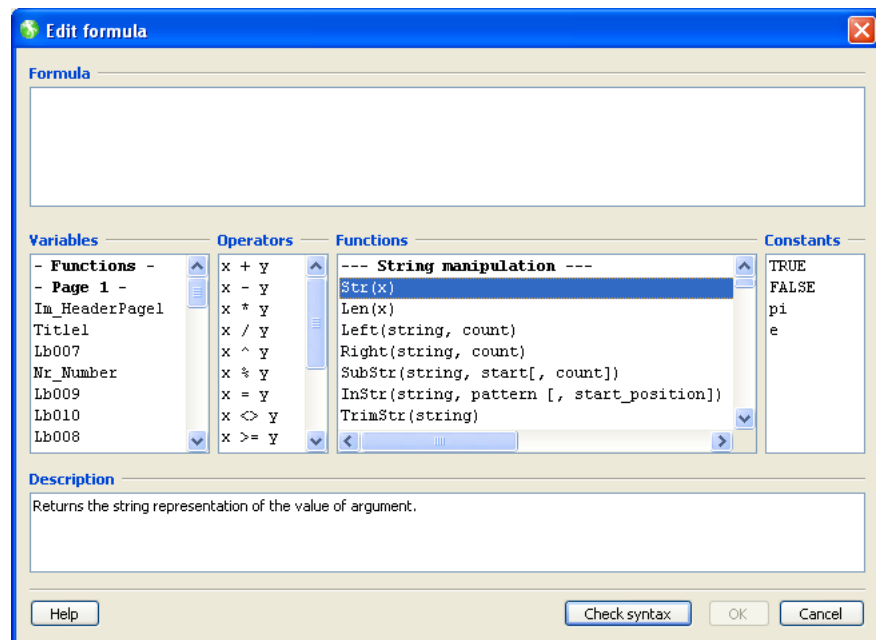
On the processing side, a suitable document and archival management system is necessary. The technical and organizational requirements depend on internal and external (legal) guidelines. The forms can again support these systems with their unique ID and the metadata.

Thanks to their XML structure, Snapform forms can be relatively easily integrated with most document management and archival systems. Also for this task, Ringler Informatik AG can provide the necessary support.

# 5 Expression language reference

This chapter describes the elements of the expression language (formula language) in detail. The starting point is the expression editor (see Fig. 5-1), where the three zones **Functions**, **Operators** and **Constants** are covered.

**Fig.5-1** *The Expression editor*



The user interface of the Expression editor is explained in section 3.2.8. Remarks on how to insert expressions into the form can be found in section 4.5. These notes are logically valid for the Function editor whose user interface is identical to the Expression editor.

**Constants** are values which are used in expressions and have the same value all over the system. The number PI is a good example of a constant.

**Operators** are applied on one or two elements. The result is returned and assigned to the target. An example for an operator is the addition symbol (+).

**Functions** are applied to their arguments. The result is returned and assigned to the target. Function results can be connected using

operators, or they can be arguments for other functions. They are also a full replacement for Label fields which used to be the only carriers for expressions in earlier versions of Snapform.

## 5.1 Constants

### 5.1.1 TRUE

**Type** Boolean

**Description** This is a Boolean constant and has the meaning of “true” (the equivalent of “it is so”). It appears as the result of a logical operator or as return value of a function.

**Note:** *This constant can also be written in lower case (**true**)*

### 5.1.2 FALSE

**Type** Boolean

**Description** This is a Boolean constant and has the meaning of “false” (the equivalent of “it is not so”). It appears as the result of a logical operator or as return value of a function.

**Note:** *This constant can also be written in lower case (**false**)*

### 5.1.3 pi

**Type** Number

**Description** This is the number Pi ( $\frac{1}{4}$ ), which corresponds to the ratio between circumference and diameter of a circle. Its value is 3.141592653589793 at the precision defined within Snapform.

## 5.1.4

### **e**

#### **Type**

Number

#### **Description**

This is the number e, the base of the natural logarithm. Its value is 2.718281828459045 at the precision defined within Snapform.

## 5.1.5

### **null**

#### **Type**

Symbol

#### **Description**

This is a symbol for non-existing or "illegal" value (such as the result of a division by 0). This constant is used internally in Snapform, but it is not accessible in the Expression editor.

## 5.2 Operators

### 5.2.1 $x + y$

**Type** numeric, string (depending on operands)

**Type of operands** x and y numeric or string.

**Description** Addition. This operator adds the two operands and returns the result.

The result depends on the types of the operands. If the operands are numeric (coming from a numeric field, for example), they are added mathematically. If they are strings (coming from a text field), they are concatenated.

**Note:** Problems may occur when an operand looks like a number, but still is a string type. In this case, the numbers are concatenated and not added up.

### 5.2.2 $x - y$

**Type** numeric

**Type of operands** x and y numeric

**Description** Subtraction. This operator subtracts the operand y from the operand x and returns the result. The result is also numeric (if both operands are numeric). Otherwise, the result is empty.

### 5.2.3 $x * y$

**Type** numeric

**Type of operands** x and y numeric

## Description

Multiplication. This operator multiplies the two operands and returns the result. The result is also numeric (if both operands are numeric). Otherwise, the result is empty.

## 5.2.4

**$x / y$**

### Type

numeric

### Type of operands

x and y numeric

## Description

Division. This operator divides the operand x by the operand y and returns the result. The result is also numeric (if both operands are numeric). Otherwise, the result is empty.

**Note:** *With a division by 0 (or an empty value), the result is empty, and there is no specific warning about the fact that subsequent results will be incorrect.*

## 5.2.5

**$x ^ y$**

### Type

numeric

### Type of operands

x and y numeric

## Description

Power. This operator sets the operand x to the y-th power and returns the result. The result is also numeric (if both operands are numeric). Otherwise, the result is empty.

## 5.2.6

**$x \% y$**

### Type

numeric

### Type of operands

x and y numeric

## Description

Remainder. This operator divides the operand x by the operand y, determines the remainder (Modulo) and returns it as the result. The result is also numeric (if both operands are numeric). Otherwise, the result is empty.

**Note:** *With a division by 0 (or an empty value), the result is "NaN" (Not a Number), and there is no specific warning about the fact that subsequent results will be incorrect.*

## 5.2.7

**x = y**

### Type

Comparison

### Type of operands

x and y boolean, numeric, string, date; must be of the same type

## Description

Compares the operands and returns the result. If the operands are equal, the result is **TRUE**, if they are different, it is **FALSE**, and if they are of different type, the result is empty.

## 5.2.8

**x != y**

### Type

Comparison

### Type of operands

x and y boolean, numeric, string, date; must be of the same type

## Description

Compares the operands and returns the result. If the operands are different, the result is **TRUE**, if they are equal, it is **FALSE**, and if they are of different type, the result is empty.

## 5.2.9

**x >= y**

### Type

Comparison



## Type of operands

x and y numeric

## Description

Compares the two operands and returns the result. If the operand x is greater than or equal to the operand y, the result is **TRUE**, if it is smaller, it is **FALSE**.

**Note:** *When operands of the string type are compared, the result is always FALSE.*

## 5.2.10

**x > y**

## Type

Comparison

## Type of operands

x and y numeric

## Description

Compares the two operands and returns the result. If the operand x is greater than the operand y, the result is **TRUE**, if it is smaller or equal, it is **FALSE**.

**Note:** *When operands of the string type are compared, the result is always FALSE.*

## 5.2.11

**x <= y**

## Type

Comparison

## Type of operands

x and y numeric

## Description

Compares the two operands and returns the result. If the operand x is smaller than or equal to the operand y, the result is **TRUE**, if it is greater, it is **FALSE**.

**Note:** *When operands of the string type are compared, the result is always FALSE.*

## 5.2.12

### **$x < y$**

#### **Type**

Comparison

#### **Type of operands**

x and y numeric

#### **Description**

Compares the two operands and returns the result. If the operand x is smaller than the operand y, the result is **TRUE**, if it is greater or equal, it is **FALSE**.

**Note:** *When operands of the string type are compared, the result is always FALSE.*

## 5.2.13

### **$x \text{ AND } y$**

#### **Type**

Logical

#### **Type of operands**

x and y boolean

#### **Description**

Logical AND: Compares the operands and returns the result. If both operands are **TRUE**, the result is **TRUE**, if one or both operands are **FALSE**, the result is **FALSE**.

The operands are normally comparison operations

## 5.2.14

### **$x \text{ OR } y$**

#### **Type**

Logical

#### **Type of operands**

x and y boolean

## Description

Logical OR: Compares the operands and returns the result. if one of or both operands are **TRUE**, the result is **TRUE**, if both operands are **FALSE**, the result is **FALSE**.

The operands are normally comparison operations

## 5.2.15

## NOT x

## Type

Logical

## Type of operands

Boolean

## Description

Logical inversion: Inverts the value of the operand and returns the result. If the operand is **TRUE**, the result is **FALSE**, if it is **FALSE**, the result is **TRUE**.

This is a single-operand operator.

## 5.3 Functions

### 5.3.1 Abs(x)

**Category**

Math function

**Short description**

Calculates the absolute value of the argument.

**Arguments**

**x**  
numeric, any value

**Result**

numeric

**Explanation**

Returns the absolute value of the argument. If it is greater than **0**, the result is identical to the argument. In the other case, the result value is the multiplication of the argument with **-1** (which is the same as the argument without minus sign). When the argument is not a number, the result of the function is **null**.

**See also**

Other Math functions

**Example**

see func\_math1.qdf

### 5.3.2 Acos(x)

**Category**

Math function

**Short description**

Returns the Arccosine of the argument in degrees.

**Arguments**

**x**  
numeric, between **-1** and **+1**

## Result

numeric, is between **0°** and **180°**

## Explanation

Returns the Arccosine of the argument in degrees. When the argument is greater than **+1** or smaller than **-1**, the calculation is not possible, and a special icon is shown in the field showing the result.

## See also

**Cos(x)**, **Asin(x)**, **Atan(x)**, **Sin(x)**, **Tan(x)**, other Math functions

## Example

see func\_math2.qdf

## 5.3.3

### **addDay(date\_value, number)**

## Category

Date function

## Short description

Calculates the date which is from the date **date\_value** away by **number** of days.

## Arguments

**date\_value**

valid date

**number**

integer

## Result

Date

## Explanation

Returns the date which is by **number** of days apart from the specified date **date\_value**. The date argument can be in the future or the past. A positive number for **number** means that the returned date is in the future from the entered data, and a negative number means that it is in the relative past.

For calculations from the current date, it is possible to use as the value for **date\_value** the function **Today()**. If the value for **number** has numbers after the decimal sign, they will be ignored.

In order to display the result, an accordingly formatted date field is necessary.

### See also

**addMonth(date\_value, number)**, **addYear(date\_value, number)**,  
**calcDays(date\_first, date\_second)**, **Date(year, month, day)**,  
**getDay(date\_value)**, **Today()**, other date functions

### Example

see func\_date.qdf

## 5.3.4

### **addHour(time\_value, number)**

### Category

Date function

### Short description

Calculates the time which is from the time **time\_value** apart by **number** of hours.

### Arguments

**time\_value**

valid time

**number**

integer

### Result

Time

### Explanation

Returns the time which is by **number** of hours apart from the entered time **time\_value**. The entered time can be in the future or the past. A positive value for **number** means that the resulting time is in the relative future from the entered time, and a negative number means that the resulting time is in the relative past from the entered time.

For calculations from the current time, it is possible to use as the value for **time\_value** the function **Now()**. If the value for **number** has numbers after the decimal sign, they will be ignored.

In this function, only hours are added up. The value for the minutes remains unchanged. If minutes must be taken into account as well, use the function **addMinute(time\_value, number)** to calculate minutes.

In order to display the result, an accordingly formatted date field is necessary (format **HH:mm**). It is possible to format the date field as a date format. In this case, the date belonging to the calculated time will be displayed.

### See also

**addMinute(time\_value, number)**, **calcHours(date\_first, date\_second)**, **getHour(time\_value)**, **Now()**, other date and time functions.

### Example

see func\_time.qdf

## 5.3.5

### **addMinute(time\_value, number)**

### Category

Date function

### Short description

Calculates the time which is from the time **time\_value** apart by **number** of minutes.

### Arguments

**time\_value**

valid time

**number**

integer

### Result

Time

### Explanation

Returns the time which is by **number** of minutes apart from the entered time **time\_value**. The entered time can be in the future or the past. A positive value for **number** means that the resulting time is in the relative future from the entered time, and a negative number means that the resulting time is in the relative past from the entered time.

For calculations from the current time, it is possible to use as the value for **time\_value** the function **Now()**. If the value for **number** has numbers after the decimal sign, they will be ignored.

In this function, only the minutes are added up; the value for the hours will be calculated accordingly.

In order to display the result, an accordingly formatted date field is necessary (format **HH:mm**). It is possible to format the date field as a date format. In this case, the date belonging to the calculated time will be displayed.

### See also

**addHour(time\_value, number)**, **calcMinutes(date\_first, date\_second)**, **getMinute(time\_value)**, **getMinute(time\_value)**, **Now()**, other date and time functions

### Example

see func\_time.qdf

## 5.3.6

### **addMonth(date\_value, number)**

### Category

Date function

### Short description

Calculates the date which is from the date **date\_value** away by **number** months.

### Arguments

**date\_value**

valid date

**number**

integer

### Result

Date

### Explanation

Returns the date which is by **number** of months apart from the specified date **date\_value**. The date argument can be in the future or the past. A



positive number for **number** means that the returned date is in the future from the entered data, and a negative number means that it is in the relative past.

For calculations from the current date, it is possible to use as the value for **date\_value** the function **Today()**. If the value for **number** has numbers after the decimal sign, they will be ignored.

In order to display the result, an accordingly formatted date field is necessary.

In this functions, only months are added up. The value for the days remains unchanged. If necessary, the value for the days can be changed using the function **addDay(date\_value, number)**.

### See also

**addDay(date\_value, number)**, **addYear(date\_value, number)**, **Date(year, month, day)**, **getMonth(date\_value)**, **Today()**, other date functions

### Example

see func\_date.qdf

## 5.3.7

### **addYear(date\_value, number)**

### Category

Date function

### Short description

Calculates the date which is from the date **date\_value** away by **number** years.

### Arguments

**date\_value**

valid date

**number**

integer

### Result

Date

## Explanation

Returns the date which is by **number** of years apart from the specified date **date\_value**. The date argument can be in the future or the past. A positive number for **number** means that the returned date is in the future from the entered data, and a negative number means that it is in the relative past.

For calculations from the current date, it is possible to use as the value for **date\_value** the function **Today()**. If the value for **number** has numbers after the decimal sign, they will be ignored.

In order to display the result, an accordingly formatted date field is necessary.

In this function, only years are added up. The value for days and months remains unchanged. If necessary the value for days and months can be changed using the function **addDay(date\_value, number)** or **addMonth(date\_value, number)**.

## See also

**addDay(date\_value, number)**, **addMonth(date\_value, number)**, **Date(year, month, day)**, **getYear(date\_value)**, **Today()**, other date functions

## Example

see func\_date.qdf

## 5.3.8

## Asin(x)

## Category

Math function

## Short description

Returns the Arcsine of the argument in degrees.

## Arguments

**x**  
numeric, between **-1** and **+1**

## Result

numeric, is between **-90°** and **+90°**

**Explanation** Returns the Arcsine of the argument in degrees. When the argument is greater than **+1** or smaller than **-1**, the calculation is not possible, and a special icon is shown in the field showing the result.

**See also** **Sin(x), Acos(x), Atan(x), Cos(x), Tan(x)**, other Math functions

**Example** see func\_math2.qdf

### 5.3.9 **Atan(x)**

**Category** Math function

**Short description** Returns the Arctangent of the argument in degrees.

**Arguments** **x**  
numeric, any value

**Result** numeric, is between **-90°** and **+90°**

**Explanation** Returns the Arctangent of the argument in degrees.

**See also** **Tan(x), Acos(x), Asin(x), Cos(x), Sin(x)**, other Math functions

**Example** see func\_math2.qdf

### 5.3.10 **calcDays(date\_first, date\_second)**

**Category** Date function

**Short description** Calculates the number of days between the entered dates.

**Arguments** **date\_first**  
valid date

## **date\_second**

valid date

### **Result**

numeric, integer

### **Explanation**

Returns the number of days between the two dates **date\_first** and **date\_second**.

The entered values can be either in the future or the past. A positive result means that **date\_second** is later than **date\_first**, a negative result means that **date\_second** is earlier than **date\_first**.

For calculations with the current date it is possible to use the value in **date\_first** or **date\_second** the function **Today()**.

### **See also**

**calcHours(date\_first, date\_second)**, **calcMinutes(date\_first, date\_second)**, **addDay(date\_value, number)**, **Date(year, month, day)**, **getDay(date\_value)**, other date functions.

### **Example**

see func\_date.qdf

## **5.3.11**

## **calcHours(date\_first, date\_second)**

### **Category**

Date function

### **Short description**

Calculates the number of hours between the entered dates.

### **Arguments**

#### **date\_first**

valid date and/or time

#### **date\_second**

valid date and/or time

### **Result**

numeric, integer

## Explanation

Returns the number of hours between the two dates **date\_first** and **date\_second**.

The entered values can be either in the future or the past. A positive result means that **date\_second** is later than **date\_first**, a negative result means that **date\_second** is earlier than **date\_first**.

For calculations with the current date it is possible to use the value in **date\_first** or **date\_second** the function **Today()**.

If the entered data is formatted as date, the same time will be used in both cases. The result is in such a case always a multiple of 24. When the entered data contains time information, the actual hours between the values are considered.

## See also

**calcDays(date\_first, date\_second)**, **calcHours(date\_first, date\_second)**, **addMinute(time\_value, number)**, **Date(year, month, day)**, **getMinute(time\_value)**, other date functions

## Example

see func\_time.qdf

## 5.3.12

### **calcMinutes(date\_first, date\_second)**

## Category

Date function

## Short description

Calculates the number of minutes between the entered dates.

## Arguments

**date\_first**

valid date and/or time

**date\_second**

valid date and/or time

## Result

numeric, integer

## Explanation

Returns the number of minutes between the two dates **date\_first** and **date\_second**.

The entered values can be either in the future or the past. A positive result means that **date\_second** is later than **date\_first**, a negative result means that **date\_second** is earlier than **date\_first**.

For calculations with the current date it is possible to use the value in **date\_first** or **date\_second** the function **Today()**.

If the entered data is formatted as date, the same time will be used in both cases. The result is in such a case always a multiple of 24. When the entered data contains time information, the actual minutes between the values are considered.

## See also

**calcDays(date\_first, date\_second), calcHours(date\_first, date\_second), addHour(time\_value, number), Date(year, month, day), getHour(time\_value)**, other date functions

## Example

see func\_time.qdf

## 5.3.13

### Ceil(x)

## Category

Math function

## Short description

Rounds the entered value to the next greater integer.

## Arguments

**x**  
numeric, any value

Result

numeric

## Explanation

Returns the entered value rounded up to the next greater integer.

## See also

**Cover(x), Round(x), Floor(x), Trunc(x)**

## Example

see func\_math1.qdf

## 5.3.14

### Cell(table, column, row)

## Category

Table function

## Short description

Represents the value of the table cell of table **table**, column **column**, row **row**.

## Arguments

### **table**

Valid name of a table

### **column**

Valid element name or column number of the table **table**

### **row**

integer

## Result

Varies depending on the format of the element for the column **column**

## Explanation

Returns the value of the table cell located in the table **table**, column **column** and row **row**.

The column **column** can either be called via its internal number (see Abschnitt 4.4.7.2) or via the name of the defining field. If this is not a valid column (such as if the field is not part of the table), the function returns **null** (corresponding to an empty value). The same is the case if the value of **row** is greater than the number of rows of the table.

Because the type of the returned value may not be predictable, make sure when defining an expression that the result or its format can be processed in further expressions.

**See also** `getCol()`, `getRow()`, `getValues(delimiter, value1 [, value2[, ...]])`

**Example** see `func_table2.qdf`

## 5.3.15 `checkCardNumber(value)`

**Category** Consistency check function

**Short description** Validates credit card numbers.

**Arguments** **value**  
number or string, corresponding to a credit card number

**Result** boolean (**TRUE** or **FALSE**)

**Explanation** Validates the formal validity of a credit card number using the *Luhn algorithm*, also known as “MOD-10 algorithm”. When the validation is successful, the result is **TRUE**, otherwise, it is **FALSE**. When the entered value is not numeric or a string, the result is **NULL**.

**See also** `checkMod10(value)`, `getMod10(value)`

**Example** see `func_consist.qdf`

## 5.3.16 `checkMod10(value)`

**Category** Consistency check function

**Short description** Validates number using the “MOD-10 algorithm”.

**Arguments** **value**  
number or string; may only contain number and space characters.



**Result** boolean (**TRUE** or **FALSE**)

**Explanation** Validates the formal validity of numbers using the *MOD-10 algorithm*. When the validation is successful, the result is **TRUE**, otherwise, it is **FALSE**. When the entered value is not numeric or a string, the result is **NULL**.

**See also** **checkCardNumber(value)**, **getMod10(value)**, **checkMod10r(value)**

**Example** see func\_consist.qdf

## 5.3.17 **checkMod10r(value)**

**Category** Consistency check function

**Short description** Validates number using the "Recursive MOD-10 algorithm".

**Arguments** **value**  
number or string; may only contain number and space characters.

**Result** boolean (**TRUE** or **FALSE**)

**Explanation** Validates the formal validity of numbers using the "*Recursive MOD-10 algorithm*". When the validation is successful, the result is **TRUE**, otherwise, it is **FALSE**. When the entered value is not numeric or a string, the result is **NULL**.

**See also** **checkMod10(value)**, **checkCardNumber(value)**, **getMod10(value)**

**Example** see func\_consist.qdf

## 5.3.18 Compress(x)

### Category

Special functions

### Short description

Creates a ZLIB compressed byte sequence from the entered value, which may be used in a barcode using the encoding method **Binary**.

### Arguments

**x**

String, byte sequence; numeric values are automatically converted to a string

### Result

Byte sequence

### Explanation

Creates a compressed byte sequence from the entered value which can be used in a 2D barcode using the encoding method **Binary**. This function is almost exclusively used with 2D barcodes.

Strings are compressed directly. Numbers are internally converted to a string and then compressed. Date values must be previously converted to a string (using the function **Str(x)**).

### See also

**getBytes(string), SubArray(byte\_array, start [, count])**

### Example

see func\_barcode.qdf

## 5.3.19 Cos(x)

### Category

Math function

### Short description

Calculates the Cosine of an angle defined in degrees.

### Arguments

**x**

numeric, any value, the entered values repeat in cycles of 360°

**Result** numeric, is between **-1** and **+1**

**Explanation** Returns the Cosine of the entered value. The entered value is assumed to be in degrees. The full circle corresponds to 360°.

**See also** **Acos(x), Asin(x), Atan(x), Sin(x), Tan(x)**, other Math functions

**Example** see func\_math2.qdf

## 5.3.20 **Cover(x)**

**Category** Math function

**Short description** Rounds the absolute value of the entered value to the next greater integer.

**Arguments** **x**  
numeric, any value

**Result** numeric

**Explanation** Returns without consideration of a minus sign the entered value rounded up to the next greater integer. This function corresponds to the equation

$$Cover(x) = \frac{x}{Abs(x)} \times Ceil(x)$$

This function is therefore for positive numbers identical with **Ceil(x)**; for negative numbers, the returned value differs from the result of **Ceil(x)** by 1.

**See also** **Ceil(x), Round(x), Floor(x), Trunc(x)**

**Example** see func\_math1.qdf

## 5.3.21

## createGUID()

### Category

Identifier function

### Short description

Creates an unique identifier (GUID, Global Unique Identifier).

### Arguments

This function has no arguments.

### Result

String

### Explanation

Creates a structured string consisting of 36 randomly selected characters. The algorithm used for this purpose ensures that this string is unique world wide (compared with other strings created in the same way).

### See also

**getRandomString(length [, digits]), getSessionID(), getPrintID(), getDataID(), getPrintIdStr(length [, digits])**

### Example

see func\_ident.qdf

## 5.3.22

## Date(year, month, day)

### Category

Date function

### Short description

Creates a date object from the arguments.

### Arguments

#### **year**

Whole positive number, corresponding to a year number

#### **month**

Whole positive number, corresponding to a month number

#### **day**

Whole positive number, corresponding to a day number

## Result

Date

## Explanation

Creates a date from the three arguments. The value for **year** must consist of four digits for current dates (entering a two-digit number leads to dates in the Roman era).

The value for **month** should be between 1 and 12. Greater values are converted and the year number increased accordingly (Example: corresponds to March of the following year).

The value for **day** should be between 1 and 28 to 31, depending on the value for **month**. Greater values are converted and accordingly added to the month and year values (example: the "35th of May" corresponds to the 4th of June).

The entries should be integers, however any digits after the decimal sign will be ignored.

## See also

**getDay(date\_value), getMonth(date\_value), getYear(date\_value),**  
and other date functions

## Example

see func\_date.qdf

## 5.3.23

## Exp(x)

## Category

Math function

## Short description

Calculates the value of the exponential function  $e^x$ .

## Arguments

**x**  
numeric, any value

## Result

numeric, greater than 0

## Explanation

Calculates the value of the exponential function  $e^x$  for the entered value and returns the result. For arguments greater than **0**, the result is greater than **1**; arguments smaller than **0** lead to results between **0** and **1**.  $e^0$  leads to the value **1**. The inverse function is **Log(x)**.

## See also

**Log(x)**

## Example

see func\_math2.qdf

## 5.3.24

### **find(string, regex[, start] )**

## Category

Regular Expression function

## Short description

Finds the position of the first occurrence of a match after position **start** of the pattern defined in **regex** within the entered value **string**.

## Arguments

**string**

String

**regex**

Regular Expression

**start** (optional)

position, integer beginning with 0

## Result

numeric, integer between 0 and length of **string** - 1, or **-1**

## Explanation

Finds the position of the first match after the starting position **start** of the pattern defined in **regex** within the entered value **string**. The result is the position, or, if there is no match, **-1**. The numbering of positions begins with 0. When the argument **start** is missing, the search for a match starts by default with position 0.

An overview over the Regular Expression syntax can be found in section 5.4.

**See also** `match(string, regex), replaceAll(orig_string, regex, replace_with)`

**Example** see func\_regex.qdf

## 5.3.25 **Floor(x)**

**Category** Math function

**Short description** Rounds the entered value to the next smaller integer.

**Arguments** **x**  
numeric, any value

**Result** numeric

**Explanation** Returns the entered value rounded down to the next smaller integer. This is (at least for positive numbers) the equivalent to cutting off the digits after the decimal sign.

**See also** `Trunc(x), Round(x), Ceil(x), Cover(x)`

**Example** see func\_math1.qdf

## 5.3.26 **getBytes(string)**

**Category** String function

**Short description** Encodes the entered character sequence to a byte sequence which can be passed to a 2D barcode using the encoding **Binary**.

**Arguments** **string**  
numeric, any value

## Result

Byte sequence

## Explanation

Converts the string passed as argument to a byte sequence which can be used to create a 2D barcode using the encoding **Binary**. The conversion uses the UTF-8 character set encoding. The byte sequence created with this function can be further compressed using the function **Compress(x)**.

**Note:** *The ISO 8859-1 character set encoding used in earlier Snapform versions is fully compatible to UTF-8, which means that there will be no incompatibilities with older Snapform forms.*

## See also

**Compress(x), SubArray(byte\_array, start [, count])**

## Example

see func\_string2.qdf and func\_barcode.qdf

## 5.3.27

## getCol()

## Category

Table function

## Short description

Shows the column number of the according field in the respective table.

## Arguments

This function has no arguments.

## Result

numeric, integer

## Explanation

Returns the number of the column in the table of the field calling the function. This value can be used in subsequent calculations. Calling this function only makes sense in connection with tables. When the calling field is not part of a table, the function returns the value **-1**.

## See also

**getRow(), Cell(table, column, row)**

## Example

see func\_table2.qdf



## 5.3.28 **getContent(url)**

### Category

Special functions

### Short description

Loads the (text) contents of **url**.

### Arguments

**url**  
valid URL

### Result

The HTML 3.2 compatible part of URL is loaded

### Explanation

This function is used to retrieve contents of URLs. The contents has to be called from a valid path **url**. This function supports most URI protocols (http:, https:, file:, ftp:, etc.).

The contents must be compatible with HTML 3.2. The primary use of this function is to load text into Text fields, Info points, or Labels; any place where HTML-formatted text can be displayed. Another purpose of the function is using webservices, as long as their response comes in a suitable format.

### See also

**importData(url)**, as well as Snapform data structure

### Example

see func\_misc.qdf

## 5.3.29 **getDataID()**

### Category

Identifier function

### Short description

Creates an unique identification string (GUID, Global Unique Identifier), which can be linked to a set of forms data.

### Arguments

This function has no arguments.

## Result

String

## Explanation

Creates a structured string consisting of 36 randomly selected characters. The algorithm used for this purpose ensures that this string is unique world wide (compared with other strings created in the same way). This identifier can, for example be used with data transfers, or in digital workflows.

## See also

**createGUID(), getSessionID(), getPrintID(), getPrintIdStr(length [, digits], getRandomString(length [, digits])**

## Example

see func\_ident.qdf

## 5.3.30

### getDay(date\_value)

## Category

Date function

## Short description

Shows the day of month of the date **date\_value**.

## Arguments

**date\_value**

valid date

## Result

Whole number between **1** and **31**

## Explanation

Returns the day of month of the entered date **date\_value**. The entered date may be in the future or in the past.

This function is part of the inverse function of **Date(year, month, day)**.

## See also

**getMonth(date\_value), getYear(date\_value), Date(year, month, day)**, other date functions

## Example

see func\_date.qdf

### 5.3.31 **getHour(time\_value)**

**Category**

Date function

**Short description**

Shows the hour number of the time **time\_value**.

**Arguments**

**time\_value**  
valid time

**Result**

Whole number between **0** and **24**

**Explanation**

Returns the hour number of the entered time **time\_value**. This function is part of the inverse function of **Time(hour, minute[, second])**.

**See also**

**getMinute(time\_value)**, **Time(hour, minute[, second])**, other date functions

**Example**

see func\_time.qdf

### 5.3.32 **getMinute(time\_value)**

**Category**

Date function

**Short description**

Shows the minute number of the time **time\_value**.

**Arguments**

**time\_value**  
valid time

**Result**

Whole number between **0** and **59**

**Explanation**

Returns the minute number of the entered time **time\_value**. This function is part of the inverse function of **Time(hour, minute[, second])**.

**See also** **getHour(time\_value), Time(hour, minute[, second])**, other date functions

**Example** see func\_time.qdf

### 5.3.33 **getMod10(value)**

**Category** Consistency check function

**Short description** Returns a checksum created with the “MOD-10 algorithm” over the entered value **value**.

**Arguments** **value**  
number or string, containing numeric or alphanumeric characters

**Result** Positive integer

**Explanation** Calculates a checksum over the entered value **value**, using the “*MOD-10 algorithm*”. The result is a integer which can, for example be used in certain 1D barcodes.

**See also** **checkMod10(value), getMod10r(value), checkMod10r(value)**

**Example** see func\_consist.qdf

### 5.3.34 **getMod10r(value)**

**Category** Consistency check function

**Short description** Returns a checksum created with the “Recursive MOD-10 algorithm” over the entered value **value**.

## Arguments

### **value**

number or string, containing numeric or alphanumeric characters

## Result

Positive integer

## Explanation

Calculates a checksum over the entered value **value**, using the “*Recursive MOD-10 algorithm*”. The result is a integer which can, for example be used in certain 1D barcodes.

## See also

**checkMod10r(value)**, **getMod10(value)**, **checkMod10(value)**

## Example

see func\_consist.qdf

## 5.3.35

### **getMonth(date\_value)**

## Category

Date function

## Short description

Shows the number of the month of the date **date\_value**.

## Arguments

### **date\_value**

valid date

## Result

Whole number between **1** and **12**

## Explanation

Returns the number of the month of the entered date **date\_value**. The entered date may be in the future or in the past. This function is part of the inverse function of **Date(year, month, day)**.

## See also

**getDay(date\_value)**, **getYear(date\_value)**, **Date(year, month, day)**, other date functions

## Example

see func\_date.qdf

## 5.3.36

### getPrintID()

#### Category

Identifier function

#### Short description

Creates an unique identification string (GUID, Global Unique Identifier), which can be linked to a print job.

#### Arguments

This function has no arguments.

#### Result

String

#### Explanation

Creates a structured string consisting of 36 randomly selected characters. The algorithm used for this purpose ensures that this string is unique world wide (compared with other strings created in the same way).

This identifier can, for example be used to identify print jobs.

#### See also

**getPrintIdStr(length [, digits], createGUID(), getSessionID(), getDataID(), getRandomString(length [, digits])**

#### Example

see func\_ident.qdf

## 5.3.37

### getPrintIdStr(length [, digits]

#### Category

Identifier function

#### Short description

Creates a freely configurable identifier string which can be linked to a print job.

#### Arguments

##### **length**

length of the string to be created; positive integer

**digits** (optional)

indicator whether the string should consist of numbers only; boolean, **FALSE** when not specified

## Result

String, alphanumeric or numeric (depending on the value of the argument **digits**)

## Explanation

Creates an unique identification string which can be linked with a print job. The length of this string is controlled with the argument **length**. The argument **digits** defines whether the string should contain number only.

## See also

**getPrintID()**, **createGUID()**, **getSessionID()**, **getDataID()**, **getRandomString(length [, digits])**

## Example

see func\_ident.qdf

## 5.3.38

### **getRandomString(length [, digits])**

## Category

Identifier function

## Short description

Creates a randomly assembled string of the length **length**.

## Arguments

**length**

length of the string to be created; positive integer

**digits** (optional)

indicator whether the string should consist of numbers only; boolean, **FALSE** when not specified

## Result

String, alphanumeric or numeric (depending on the value of the argument **digits**)

## Explanation

Creates a randomly assembled string. The length of the string is controlled with the argument **length**. The argument **digits** defines whether the string contains numbers only. This function can also be used (when the value of **digits** **TRUE**) to create a random number.

## See also

**getPrintIdStr(length [, digits], createGUID(), getSessionID(), getPrintID(), getDataID())**

## Example

see func\_ident.qdf

## 5.3.39

### getRow()

## Category

Table function

## Short description

Shows the row number of the according field in the respective table.

## Arguments

This function has no arguments.

## Result

numeric, integer

## Explanation

Returns the number of the row in the table of the field calling the function. This value can be used in subsequent calculations.

Calling this function only makes sense in connection with tables. When the calling field is not part of a table, the function returns the value **-1**.

## See also

**getCol(), Cell(table, column, row)**

## Example

see func\_table2.qdf

## 5.3.40

### getSelectedIndex(ComboBox)

## Category

Special functions



<b>Short description</b>	Returns the index number of the current selection of a List field.
<b>Arguments</b>	<b>ComboBox</b> Name of a selection list field
<b>Result</b>	numeric, integer between 1 and the number of entries of the list field
<b>Explanation</b>	Returns the index number of the current selection of a list field. Numbering starts with 1.
<b>See also</b>	<b>setList(ComboBox, str_items, str_values),</b> <b>setSelectedIndex(ComboBox, index)</b>
<b>Example</b>	see func_combobox.qdf

## 5.3.41 getSessionID()

<b>Category</b>	Identifier function
<b>Short description</b>	Creates an unique identification string (GUID, Global Unique Identifier), which can be assigned to the current working session with Snapform Viewer.
<b>Arguments</b>	This function has no arguments.
<b>Result</b>	String
<b>Explanation</b>	<p>Creates a structured string consisting of 36 randomly selected characters. The algorithm used for this purpose ensures that this string is unique world wide (compared with other strings created in the same way).</p> <p>This identifier is used to identify the Snapform Viewer working session, and it remains constant for the whole length of the session.</p>

**See also** `createGUID()`, `getPrintID()`, `getDataID()`, `getRandomString(length [, digits])`

**Example** see `func_ident.qdf`

## 5.3.42 `getValues(delimiter, value1 [, value2[, ...]])`

**Category** Table function

**Short description** Assembles the values of the indicated fields to a string, using the delimiter character **delimiter**.

**Arguments**

**delimiter**  
delimiting character; must be entered between double quotes.

**value1 ... valuen**  
Name of the fields whose values are to be assembled to form the result.

**Result** String

**Explanation** Assembles the values of the fields specified as **value1 ... valuen** to a string. The delimiting character is **delimiter**. This function is primarily used for preparing data streams for 2D barcodes, but it can also be used for data transfers. The field names must not be part of a table. Values of fields which are assigned to table elements will not be added to the result. To read out values from tables, use the function **TblValues(table, tbl\_start, tbl\_end, row\_start, row\_end, delimiter, column [, column[, ...]])**. Field values are represented as strings, even if they were originally defined as numbers or dates.

**See also** `TblValues(table, tbl_start, tbl_end, row_start, row_end, delimiter, column [, column[, ...]])`

**Example** see `func_table2.qdf`

### 5.3.43 **getVersion()**

#### Category

Special functions

#### Short description

Returns the version number of the running Snapform application (Designer, Viewer).

#### Arguments

This function has no arguments.

#### Result

String

#### Explanation

Returns the version number of the running Snapform application (Designer, Viewer). This number can be used to catch potential compatibility issues between different Snapform versions.

#### See also

no related functions

#### Example

see func\_ident.qdf

### 5.3.44 **getYear(date\_value)**

#### Category

Date function

#### Short description

Shows the number of the year of the date **date\_value**.

#### Arguments

**date\_value**  
valid date

#### Result

Whole number

#### Explanation

Returns the number of the year of the entered date **date\_value**. The entered date may be in the future or in the past.

This function is part of the inverse function of **Date(year, month, day)**.

### See also

**getDay(date\_value)**, **getMonth(date\_value)**, **Date(year, month, day)**, other date functions

### Example

see func\_date.qdf

## 5.3.45

### **if(logical\_test, value\_if\_true [,value\_if\_false])**

### Category

Special functions

### Short description

Executes a comparison operation on the specified operands and returns a value depending on its result.

### Arguments

#### **logical\_test**

comparison operation

#### **value\_if\_true**

Return value if the comparison operation returns **TRUE** ; the format of the return value is determined by the value itself.

#### **value\_if\_false**

Return value if the comparison operation returns **FALSE**; the format of the return value is determined by the value itself.

### Result

numeric

### Explanation

This function executes a comparison operation on the operands (which can be expressions themselves) and returns if its result is **TRUE** the value **value\_if\_true**. When the comparison results in **FALSE** the function returns the value **value\_if\_false**. These return values can be expressions as well.

The type of the return values **value\_if\_true** and **value\_if\_false** should be the same (string or numeric or boolean), in order to avoid potential conflicts with field types.

**See also**

Comparison operators (section 5.2.7 to section 5.2.14)

**Example**

see func\_misc.qdf

**5.3.46**

**importData(url)**

**Category**

Special functions

**Short description**

Imports a data file from **url**.

**Arguments**

**url**  
valid URL

**Result**

The field name/field value pairs defined in the imported file are loaded into the form

**Explanation**

This function is used to import data from external sources (URLs). These files have to be called from a valid path **url**. This function supports essentially all URI protocols (http:, https:, file:, ftp:, etc.).

The import files must either be in the QDF/A format, or as XML using the Snapform data structure. Field name/field value pairs in the import file are loaded to the according fields of the form. Field name/field value pairs which have no according field in the form are ignored. Files which have a non-conforming structure are ignored as well.

**See also**

**getContent(url)**, as well as Snapform data structure

**Example**

see func\_misc.qdf

## 5.3.47 IncVersion(initial\_value, step)

### Category

Math function

### Short description

Loop counter: sets a counter to **initial\_value** and increments it with each subsequent call by **step**.

### Arguments

**initial\_value**

numeric

**step**

numeric

### Result

numeric

### Explanation

This function returns, when it is called for the first time, the initial value **initial\_value** and keeps it stored. With every subsequent call, this value is incremented by the value **step**, stored and returned.

This function can be used as a loop counter. In this case, both arguments are integers.

### See also

no related functions

### Example

see func\_math1.qdf

## 5.3.48 InStr(string, pattern [, start\_position])

### Category

String function

### Short description

Returns the position of **pattern** in the string **string**, beginning with the search from position **start\_position**.

## Arguments

### **string**

string to be searched, or field name

### **pattern**

string which should be found

### **start\_position**

position in string **string**, from which the search should start; optional, integer, greater than or equal to **0**; default value when it is not specified is **0**.

## Result

numeric, integer greater than or equal to **0**; **-1** if **pattern** is not contained in **string**

## Explanation

Searches the string **string** for the string **pattern** and returns the position at which **pattern** begins.

When a string is entered for **pattern**, it must be placed between double quotes.

With **start\_position** the position from which on the search should start. This can be useful when the search for another occurrence of **pattern** than the first one is intended.

The position numbers are 0-based. Therefore, the position of the first character is **0**. When **pattern** is not found, the function returns **-1**.

The search is precise, which means that capitalization is taken into account. This fact must be considered when assembling the expression.

## See also

**Left(string, count)**, **Right(string, count)**, **SubStr(string, start[, count])** and the other string functions

## Example

see func\_string2.qdf

## 5.3.49 **isEmpty(x)**

### Category

Special functions

### Short description

Tests whether the argument **x** is empty (or **null**).

### Arguments

**x**  
Field name or result of another calculation

### Result

boolean; **TRUE** when **x** is empty, otherwise **FALSE**

### Explanation

Tests whether the argument contains the empty value (or null). This can be, for example, useful when the completeness of the form is verified, or if catching erroneous results, when it is possible that not allowed values may be present (such as when a division by 0 is possible).

### See also

no related functions

### Example

see func\_misc.qdf

## 5.3.50 **Left(string, count)**

### Category

String function

### Short description

Returns the first **count** characters, counted from left, of the string **string**.

### Arguments

**string**  
string or name of a text field

**count**  
number of characters to be returned; positive integer



**Result** String or empty value (when **string** has fewer characters than specified in **count**)

**Explanation** Returns the first **count** characters of the string **string**. The characters are counted from the left hand side.

This function returns only a valid value when **string** has a length of at least **count** characters. When **string** is shorter, the empty value is returned.

**See also** **Right(string, count)**, **SubStr(string, start[, count])** and other string functions

**Example** see func\_string1.qdf

## 5.3.51 Len(x)

**Category** String function

**Short description** Returns the length (number of characters) of the string.

**Arguments** **x**  
String

**Result** numeric; positive integer or empty value (when **x** is not a string)

**Explanation** Returns the length (number of characters) of the string **x**.

**See also** **Str(x)**, **InStr(string, pattern [, start\_position])**, **Left(string, count)**, **Right(string, count)** and other string functions

**Example** see func\_string1.qdf

## 5.3.52

## Log(x)

### Category

Math function

### Short description

Calculates the Natural Logarithm of x (ln x).

### Arguments

**x**  
numeric, positive number

### Result

numeric

### Explanation

Calculates the Natural Logarithm (base **e**) of **x** and returns it.

**x** must be greater than **0**. If **x** is equal to **0** the result is  $-\infty$ . When **x** is smaller than **0**, the logarithm is not defined, and the error symbol gets displayed.

Frequently, the logarithm to base 10 is used. It is calculated as follows:

$$= \text{Log}(\mathbf{x}) / \text{Log}(10)$$

and similarly the logarithm to base 2:

$$= \text{Log}(\mathbf{x}) / \text{Log}(2)$$

The inverse function is **Exp(x)**.

### See also

**Exp(x)**

### Example

see func\_math2.qdf

## 5.3.53

## match(string, regex)

### Category

Regular Expression function

**Short description** Determines whether the pattern defined in **regex** is contained in the entered value **string**.

**Arguments**

**string**  
String

**regex**  
Regular Expression

**Result** boolean (**TRUE** or **FALSE**)

**Explanation**

Determines whether the entry value string contains a match of the pattern defined in the Regular Expression regex.

An overview over the Regular Expression syntax can be found in section 5.4.

**See also** **find(string, regex[, start] ), replaceAll(orig\_string, regex, replace\_with)**

**Example** see func\_regex.qdf

## 5.3.54 **Max(x, y)**

**Category** Math function

**Short description** Determines the greater of the two values.

**Arguments**

**x**  
numeric, any value

**y**  
numeric, any value

**Result** numeric

**Explanation** Determines the greater of the two values **x** and **y** and returns it. When both values are equal, that value is returned.

**See also** **Min(x, y)**

**Example** see func\_math1.qdf

## 5.3.55 **Min(x, y)**

**Category** Math function

**Short description** Determines the smaller of the two values.

**Arguments**

**x**  
numeric, any value

**y**  
numeric, any value

**Result** numeric

**Explanation** Determines the smaller of the two values **x** and **y** and returns it. When both values are equal, that value is returned.

**See also** **Max(x, y)**

## 5.3.56 **Now()**

**Category** Date function

**Short description** Returns the current time.

**Arguments** This function has no arguments.

## Result

Time

## Explanation

Returns the current time. In order to display it, a date field formatted as time is necessary.

## See also

**getHour(time\_value)**, **getMinute(time\_value)** and the other time-related date functions

## Example

see func\_math1.qdf

## 5.3.57

## **replaceAll(orig\_string, regex, replace\_with)**

## Category

Regular Expression function

## Short description

Replaces all matches of the pattern defined in **regex** in the entered value **orig\_string** with the string **replace\_with**.

## Arguments

**orig\_string**

String

**regex**

Regular Expression

**replace\_with**

String

## Result

String

## Explanation

Replaces all occurrences of the pattern defined in **regex** in the entered value **orig\_string** with the string **replace\_with**. When there is no match, the entered value is returned unchanged.

An overview over the Regular Expression syntax can be found in section 5.4.

**See also** `find(string, regex[, start] )`, `match(string, regex)`

**Example** see `func_regex.qdf`

## 5.3.58 **Right(string, count)**

**Category** String function

**Short description** Returns the first **count** characters, counted from the right, of the string **string**.

**Arguments**

**string**  
string or name of a text field

**count**  
number of characters to be returned; positive integer

**Result** String or empty value (when **string** has fewer characters than specified in **count**)

**Explanation** Returns the first **count** characters, counted from the right hand side, of the string **string**. This would be the last **count** characters if counting would be from the left hand side.

This function returns only a valid value when **string** has a length of at least **count** characters. When **string** is shorter, the empty value is returned.

**See also** `Left(string, count)`, `SubStr(string, start[, count])` and other string functions

**Example** see `func_string1.qdf`

## 5.3.59 **Round(x)**

**Category** Math function

**Short description** Rounds the entered value up or down to the next whole number.

**Arguments** **x**  
numeric, any value

**Result** numeric

**Explanation** Rounds the entered value up or down to the closest whole number. When the values after the decimal sign of x is smaller than x.5, it gets rounded down, when it is greater than or equal to x.5, it gets rounded up. If the rounding should occur to the value precision,

**= precision \* Round(x/precision)**

is used

**See also** **Ceil(x), Floor(x)**

**Example** see func\_math1.qdf

## 5.3.60 **setFieldPrintable(true|false, field\_name)**

**Category** Status function

**Short description** Sets the **printable** property of the field.

**Arguments** **true|false**  
status value; **true** or **false**  
**field\_name**  
field name

## Result

Field property is set

## Explanation

Sets the **printable** property of the field **field\_name** to **true** or **false**. This corresponds to the checking or unchecking of the **printable** checkbox in the Properties dialog of the according field.

## See also

**setFieldVisible(true|false, field\_name), setFieldWritable(true|false, field\_name)**

## Example

see func\_status.qdf

## 5.3.61

**setFieldVisible(true|false, field\_name)**

## Category

Status function

## Short description

Sets the **visible** property of the field.

## Arguments

**true|false**  
status value; **true** or **false**

**field\_name**  
field name

## Result

Field property is set

## Explanation

Sets the **visible** property of the field **field\_name** to **true** or **false**. This corresponds to the checking or unchecking of the **visible** checkbox in the Properties dialog of the according field.

## See also

**setFieldPrintable(true|false, field\_name),  
setFieldWritable(true|false, field\_name)**

## Example

see func\_status.qdf



## 5.3.62 **setFieldWritable(true|false, field\_name)**

### Category

Status function

### Short description

Controls the **read-only** property of the field.

### Arguments

**true|false**

status value; **true** or **false**

**field\_name**

field name

### Result

Field property is set

### Explanation

Sets the **read-only** property of the field **field\_name** to **true** or **false**. **This corresponds to the unchecking or checking of the** read-only checkbox in the Properties dialog of the according field.

This function acts inverse to the checking and unchecking of the **read-only** checkbox. In order to check this box, the status **false** must be set in this function, and vice-versa.

### See also

**setFieldPrintable(true|false, field\_name),**  
**setFieldVisible(true|false, field\_name)**

### Example

see func\_status.qdf

## 5.3.63 **setList(ComboBox, str\_items, str\_values)**

### Category

Special functions

### Short description

Configures the selection list for the list field **ComboBox**.

## Arguments

### ComboBox

Name of a selection list field

#### **str\_items**

String with the list of the elements for the list field, enclosed with double quotes and separated with "|"

#### **str\_values**

String with the list of the return values for the list field, enclosed with double quotes and separated with "|"

## Result

The list of selections gets configured

## Explanation

This function is used to configure the selection list for the List field **ComboBox**. Applying this function corresponds to filling out the table in the configuration window of the selection list (see section 4.4.4, Fig. 4-99).

In the **str\_items** string, the entries of the left hand column (Elements) of the table are defined. The elements are separated with the | character.

In the **str\_values** string, the entries of the right hand column (values) of the table are defined. The elements are separated with the | character. When this string is missing or empty, no return values are defined for the selection list (and the List field returns the element names).

It is important to make sure that the individual elements in **str\_items** and in **str\_values** correspond to each other. If this is not the case, there will be mis-matches between element name and return value.

This function allows you to dynamically assemble selection lists. It is, for example, possible to transfer table contents into a selection list for further usage.

## See also

### **getSelectedIndex(ComboBox)**

## Example

see func\_combobox.qdf

## 5.3.64 **setPagePrintable(true|false, page\_number [, page\_number[,...]])**

**Category** Status function

**Short description** Sets the **printable** property of the affected page(s).

**Arguments**

**true|false**  
status value; **true** or **false**

**page\_number**  
page number or group of page numbers, separated with commas

**Result** The page property is set

**Explanation** Sets the **printable** property of one or several pages **page\_number** to **true** or **false**. This corresponds to the checking or unchecking of the **printable** checkbox in the Properties dialog of the affected page (see Abschnitt 4.3.4.2, Fig. 4-18).

**See also** **setPageVisible(true|false, page\_number [, page\_number[, ...]])**

**Example** see func\_status.qdf

## 5.3.65 **setPageVisible(true|false, page\_number [, page\_number[, ...]])**

**Category** Status function

**Short description** Sets the **visible** property of the affected page(s).

## Arguments

**true|false**

status value; **true** or **false**

**page\_number**

page number or group of page numbers, separated with commas

## Result

The page property is set

## Explanation

Sets the **visible** property of one or several pages **page\_number** to **true** or **false**. This corresponds to the checking or unchecking of the **visible** checkbox in the Properties dialog of the affected page (see Abschnitt 4.3.4.2, Fig. 4-18).

## See also

**setPagePrintable(true|false, page\_number [, page\_number[,...]])**

## Example

see func\_status.qdf

## 5.3.66

**setPrintable(category\_list, true|false)**

## Category

Status function

## Short description

Sets the **printable** property of the affected category(s).

## Arguments

**category\_list**

String with a list of comma-separated categories

**true|false**

status value; **true** or **false**

## Result

The field property for the fields of the according category is set

## Explanation

Sets the **printable** property of the fields belonging to the categories specified in **category\_list** to **true** or **false**. This corresponds to a

simultaneous checking or unchecking of the **printable** checkbox in the Properties dialog of all those fields.

### See also

**setVisible(category\_list, true|false), setWritable(category\_list, true|false)**

### Example

see func\_status.qdf

## 5.3.67

### **setSelectedIndex(ComboBox, index)**

### Category

Special functions

### Short description

Sets the current selection of the List field **ComboBox** to the value associated to index **index**.

### Arguments

#### **ComboBox**

Name of a selection list field

#### **index**

Index of the selection from the List field to become active; integer between 1 and the number of selectable items of the List field.

### Result

The item with Index **index** gets selected

### Explanation

Sets the current selection of the List field **ComboBox** to the item associated to index **index**.

### See also

**getSelectedIndex(ComboBox), setList(ComboBox, str\_items, str\_values)**

### Example

see func\_combobox.qdf

## 5.3.68 **setVisible(category\_list, true|false)**

### Category

Status function

### Short description

Sets the **visible** property of the affected category(s).

### Arguments

**category\_list**

String with a list of comma-separated categories

**true|false**

status value; **true** or **false**

### Result

The field property for the fields of the according category is set

### Explanation

Sets the **visible** property of the fields belonging to the categories specified in **category\_list** to **true** or **false**. This corresponds to a simultaneous checking or unchecking of the **visible** checkbox in the Properties dialog of all those fields.

### See also

**setPrintable(category\_list, true|false)**, **setWritable(category\_list, true|false)**

### Example

see func\_status.qdf

## 5.3.69 **setWritable(category\_list, true|false)**

### Category

Status function

### Short description

Controls the **read-only** property of the according category(s).

### Arguments

**category\_list**

String with a list of comma-separated categories

**true|false**  
status value; **true** or **false**

## Result

The field property for the fields of the according category is set

## Explanation

Sets the **read-only** property of the fields belonging to the categories specified in **category\_list** to **true** or **false**. This corresponds to the simultaneous checking or unchecking of the **read-only** checkbox in the Properties dialog of all those fields. This function acts inverse to the checking and unchecking of the **read-only** checkbox.

## See also

**setPrintable(category\_list, true|false), setVisible(category\_list, true|false)**

## Example

see func\_status.qdf

## 5.3.70

### Sin(x)

## Category

Math function

## Short description

Calculates the Sine of an angle defined in degrees.

## Arguments

**x**  
numeric, any value, the entered values repeat in cycles of 360°

## Result

numeric, is between **-1** and **+1**

## Explanation

Returns the Sine of the entered value. The entered value is assumed to be in degrees. The full circle corresponds to 360°.

## See also

**Asin(x), Acos(x), Atan(x), Cos(x), Tan(x)**, other Math functions

## Example

see func\_math2.qdf

## 5.3.71

## Sqrt(x)

### Category

Math function

### Short description

Calculates the square root of the entered value.

### Arguments

**x**  
numeric, value greater than 0

### Result

numeric

### Explanation

Returns the square root of the entered value. If the entered value is negative, the square root is not defined, and the error symbol is shown.

### See also

Operator **x ^ y**

### Example

see func\_math2.qdf

## 5.3.72

## Str(x)

### Category

String function

### Short description

Converts the entered value to a string.

### Arguments

**x**  
String, numeric, date, boolean

### Result

String

### Explanation

Returns the entered value represented as string.



## See also

**Len(x), SubStr(string, start[, count])**

## Example

see func\_string1.qdf

## 5.3.73

### **StrTotal(table, column)**

## Category

Table function

## Short description

Assembles the table values of column **column** to a string.

## Arguments

### **table**

Table name

### **column**

Column number or field name of the column from which the values are to be assembled to a string

## Result

String

## Explanation

Concatenates the field values of the column **column** in table **table** to a string. The "|" character is used as separator.

## See also

**TblValues(table, tbl\_start, tbl\_end, row\_start, row\_end, delimiter, column [, column[, ...]]), Total(table, column), TimeTotal(table, column)**

## Example

see func\_table1.qdf

## 5.3.74 SubArray(byte\_array, start [, count])

### Category

String function

### Short description

Creates a sub-sequence of the byte sequence **byte\_array**, beginning with byte number **start** and the number of **count** bytes.

### Arguments

#### **byte\_array**

byte sequence, normally created with **Compress(x)** or **getBytes(string)**

#### **start**

integer, between **0** and the length of the byte sequence **byte\_array**

#### **count** (optional)

Integer , between 1 and the length of the byte sequence **byte\_array**; when the value is not present, the extraction occurs to the end of the byte sequence

### Result

Byte sequence

### Explanation

This function is the equivalent of the **SubStr(string, start[, count])** function for strings, but for byte sequences.

The function creates a sub-byte sequence of the **byte\_array** byte sequence, beginning with the character at position **start** and with the length **count**. When **count** is not defined, the extraction occurs until the end of the byte sequence. The starting position of the byte sequence begins with **0**.

### See also

**getBytes(string), Compress(x)**

### Example

see func\_string2.qdf and func\_barcode.qdf

## 5.3.75 SubStr(string, start[, count])

### Category

String function

### Short description

Creates a substring, beginning at character position **start** and with length **count**.

### Arguments

#### **string**

String, entered directly, or as a result of an expression

#### **start**

Character position from which on the substring is created; integer between **0** and the length of **string**

#### **count** (optional)

Number of characters of the substring; integer between **1** and the length of **string**

### Result

String

### Explanation

Creates a substring of string with the length **count**, beginning from position **start**.

The position numbering begins with **0**. With **0** as starting value, the function is equivalent to the **Left(string, count)** function.

When no value is specified for **count**, the substring is created until the end of **string** (which is then equivalent to the **Right(string, count)** function).

### See also

**Right(string, count)**, **Left(string, count)**, **Str(x)**

### Example

see func\_string1.qdf

## 5.3.76

## Tan(x)

### Category

Math function

### Short description

Calculates the Tangent of an angle defined in degrees.

### Arguments

**x**

numeric, any value, the entered values repeat in cycles of 180°

### Result

numeric

### Explanation

Returns the Tangent of the entered value. The entered value is assumed to be in degrees. The full circle corresponds to 360°.

### See also

**Atan(x), Asin(x), Acos(x), Cos(x), Sin(x)**, other Math functions

### Example

see func\_math2.qdf

## 5.3.77

## TblValues(table, tbl\_start, tbl\_end, row\_start, row\_end, delimiter, column [, column[, ...]])

### Category

Table function

### Short description

Creates a string with table values assembled row-by-row and specific delimiters.

### Arguments

**table**

Table name

**tbl\_start**

String marking the beginning of the table

**tbl\_end**

String marking the end of the table

**row\_start**

String to mark the beginning of a table row

**row\_end**

String to mark the end of a table row

**delimiter**

String to separate the field values within a table row

**column** (repeatable)

Column number or column name in the table

**Result**

String with a structure defined by the function arguments

**Explanation**

This function is used to export the contents of a table for data transfers, or for the preparation of a 2D barcode. The possibility to define strings marking the beginning and end of the table and of table rows also allows to export the table in an XML format.

The markers and the **delimiter** may consist of one or several characters. It is also possible to use an empty string or no character at all. Quotes and the backslash character must be escaped with the Backslash character (\). When specifying a new line character (for example for row\_end), the sequence **\n** must be entered. The columns can be in any selectable order. It is even possible to repeat columns.

This function does not return column sums. These must be added separately.

**See also**

**Total(table, column), StrTotal(table, column), TimeTotal(table, column)**

**Example**

see func\_table1.qdf

## 5.3.78 **Time(hour, minute[, second])**

### Category

Date function

### Short description

Creates a time object from the arguments.

### Arguments

#### **hour**

Integer, corresponding to the hours number

#### **minute**

Integer, corresponding to the minutes number

#### **second** (optional)

Integer, corresponding to the seconds number

### Result

Time

### Explanation

Creates a time value from the three arguments. The value for **hour** should be between 0 and 24.

The value for **minute** should be between 0 and 59. Higher values are converted and the hours number is increased accordingly.

The value for **second** is optional and should be between 1 and 59. Higher values are converted and accordingly added to the minutes and hours.

The entries should be integers, however any digits after the decimal sign will be ignored.

### See also

**getHour(time\_value)**, **getMinute(time\_value)**, other date functions

### Example

see func\_time.qdf

## 5.3.79 TimeTotal(table, column)

### Category

Table function

### Short description

Adds the values of the **column** column to a time.

### Arguments

#### **table**

Table name

#### **column**

Column number or field name of the column from which the values are to be summed up

### Result

Time

### Explanation

sums the field values of the **column** column up as a time and returns the result. It is important to make sure that the **column** column is actually formatted as time.

### See also

**Total(table, column)**, **StrTotal(table, column)**

### Example

see func\_table2.qdf

## 5.3.80 Today()

### Category

Date function

### Short description

Returns the current date.

### Arguments

This function has no arguments.

### Result

Date

**Explanation** Returns the current date. In order to display it, a date field formatted as date is necessary.

**See also** **getDay(date\_value), getMonth(date\_value), getYear(date\_value)**  
and other date-oriented date functions

**Example** see func\_date.qdf

## 5.3.81 **Total(table, column)**

**Category** Table function

**Short description** Sums up the values of the **column** column.

**Arguments**

**table**  
Table name

**column**  
Column number or field name of the column from which the values are to be summed up

**Result** Number

**Explanation** sums the field values of the **column** up, and returns the resulting sum. It is important to make sure that the **column** column is actually a Number field.

**See also** **TimeTotal(table, column), StrTotal(table, column)**

**Example** see func\_table1.qdf



## 5.3.82 TrimStr(string)

### Category

String function

### Short description

Removes preceding and trailing spaces from a string.

### Arguments

#### **string**

String; may have preceding and trailing spaces

### Result

String

### Explanation

Removes preceding and trailing spaces from the entered string and returns the trimmed string. When the string does not have any preceding or trailing spaces, the result is identical with the entered string. This function is particularly useful when preparing data for 2D barcodes.

### See also

**SubStr(string, start[, count]), InStr(string, pattern [, start\_position])**

### Example

see func\_string2.qdf

## 5.3.83 Trunc(x)

### Category

Math function

### Short description

Rounds the absolute value of the entered value to the next smaller whole number.

### Arguments

#### **x**

numeric, any value

### Result

numeric

## Explanation

Returns without consideration of a minus sign, the entered value rounded down to the next smaller whole number. This function corresponds to the equation

$$Trunc(x) = \frac{x}{Abs(x)} \times Floor(x)$$

This function is therefore for positive numbers identical with **Floor(x)**; for negative numbers the returned value differs from the result of **Floor(x)** by 1.

## See also

**Floor(x), Round(x), Ceil(x), Cover(x)**

## Example

see func\_math1.qdf

## 5.4 Regular Expressions

Regular Expressions are a very powerful tool to evaluate strings. They allow finding of patterns which follow certain rules. They also allow replacing the matches with other strings, which makes them a very powerful "Search and Replace" function.

The following table is based on the software development documentation and contains the most important syntax elements for building Regular Expressions. For a more in-depth discussion of Regular Expressions, we recommend the book by Jeffrey Friedl:

Mastering Regular Expressions, 2nd Edition, Jeffrey E. F. Friedl, O'Reilly and Associates, 2002 ([Amazon Link](#)).

### 5.4.1 Overview table

Syntax element	Matches
<b>Character</b>	
<code>x</code>	The character <code>x</code>
<code>\\</code>	The Backslash <code>\</code> character
<code>\on</code>	The character with Octal value <code>on</code> ( $0 \leq n \leq 7$ )
<code>\onn</code>	The character with Octal value <code>onn</code> ( $0 \leq n \leq 7$ )
<code>\omnn</code>	The character with Octal value <code>omnn</code> ( $0 \leq m \leq 3, 0 \leq n \leq 7$ )
<code>\xhh</code>	The character with Hexadecimal value <code>0xhh</code>
<code>\uhhhh</code>	The character with Unicode (Hexadecimal) value <code>0xhhhh</code>
<code>\t</code>	The Tab character ( <code>'\u0009'</code> )
<code>\n</code>	The Line Feed character ( <code>'\u000A'</code> )
<code>\r</code>	The Carriage Return character ( <code>'\u000D'</code> )
<code>\f</code>	The Form Feed character ( <code>'\u000C'</code> )
<code>\a</code>	The Bell character ( <code>'\u0007'</code> )
<code>\e</code>	The Escape character ( <code>'\u001B'</code> )
<code>\cx</code>	The Control character, entsprechen <code>x</code>
<b>Character groups</b>	
<code>[abc]</code>	<code>a</code> , <code>b</code> , or <code>c</code> (simple Group)
<code>[^abc]</code>	Any character, except <code>a</code> , <code>b</code> , or <code>c</code> (Negation)
<code>[a-zA-Z]</code>	<code>a</code> to <code>z</code> or <code>A</code> to <code>Z</code> (Range)
<code>[a-d[m-p]]</code>	<code>a</code> to <code>d</code> , or <code>m</code> to <code>p</code> : <code>[a-dm-p]</code> (Union)

[a-z&&[def]]	d, e, or f (Intersection)
[a-z&&[^bc]]	a to z, except b and c: [ad-z] (Subtraction)
[a-z&&[^m-p]]	a to z, but not m to p: [a-1q-z] (Subtraction)

### Pre-defined character groups

.	Any character (with or without end of line)
\d	A number: [0-9]
\D	Not a number: [^0-9]
\s	A Whitespace character: [ \t\n\x0B\f\r]
\S	Not a Whitespace character: [^\s]
\w	A Word character: [a-zA-Z_0-9]
\W	Not a word character: [^\w]

### POSIX charactergroups (for US-ASCII only)

\p{Lower}	A lower case letter: [a-z]
\p{Upper}	An upper case letter: [A-Z]
\p{ASCII}	Any ASCII value: [\x00-\x7F]
\p{Alpha}	A letter: [\p{Lower}\p{Upper}]
\p{Digit}	A number: [0-9]
\p{Alnum}	An alphanumeric character: [\p{Alpha}\p{Digit}]
\p{Punct}	A Punctuation !"#%&'()*+,-./:;<=>?@[\\]^_`{ }~
\p{Graph}	A visible character [\p{Alnum}\p{Punct}]
\p{Print}	A printable character: [\p{Graph}]
\p{Blank}	Space or Tab character: [ \t]
\p{Cntrl}	A Control character [\x00-\x1F\x7F]
\p{XDigit}	A Hexadecimal character: [0-9a-fA-F]
\p{Space}	A Whitespace character [ \t\n\x0B\f\r]

### Character groups for Unicode blocks and categories

\p{InGreek}	A character from the Greek block (simple block)
\p{Lu}	An upper case letter (simple category)
\p{Sc}	A currency symbol
\P{InGreek}	A character outside of the Greek block (Negation)
[p{L}&&[^p{Lu}]]	Any character except upper case letter (Subtraction)

### Delimiters

^	Begin of a line
\$	End of a line
\b	Word delimiter
\B	Not a word delimiter
\A	Begin of input
\G	End of previous match
\Z	End of input, except the last end character, if present
\z	End of input

### Greedy multipliers

$X?$	$X$ , one or zero times
$X^*$	$X$ , zero or any times
$X^+$	$X$ , one or multiple times
$X\{n\}$	$X$ , exactly $n$ times
$X\{n, \}$	$X$ , at least $n$ times
$X\{n, m\}$	$X$ , at least $n$ but at most $m$ times

### Non-greedy multipliers

$X??$	$X$ , one or zero times
$X^*?$	$X$ , zero or any times
$X^+?$	$X$ , one or multiple times
$X\{n\}?$	$X$ , exactly $n$ times
$X\{n, \}?$	$X$ , at least $n$ times
$X\{n, m\}?$	$X$ , at least $n$ but at most $m$ times

### Possessive multipliers

$X?+$	$X$ , one or zero times
$X^*+$	$X$ , zero or any times
$X^{++}$	$X$ , one or multiple times
$X\{n\}^+$	$X$ , exactly $n$ times
$X\{n, \}^+$	$X$ , at least $n$ times
$X\{n, m\}^+$	$X$ , at least $n$ but at most $m$ times

### Logical operators

$XY$	$X$ followed by $Y$
$X Y$	Either $X$ or $Y$
$(X)$	$X$ , as a saved group

### Backward references

$\backslash n$	Match of the $n$ -th saved group
----------------	----------------------------------

**Helpful hint:** The backslash character (“\”) is used to display protected expressions, as shown in the table above, as well as for displaying characters which otherwise would be considered to be control characters (such as the backslash character itself).

# 6 Snapform File Formats

## 6.1 Form, compressed (QDF/S)

**File extension** .qdf

**Explanation** Compact Snapform format. The whole content is compressed and encrypted. This format requires special tools to read the data including metadata.

**Format description** Pure binary format. Data can be read only via API (for a format description, see the documentation for the API).

## 6.2 Form, open (QDF/A)

**File extension** .qdf

**Explanation** Open Snapform format. Only the form description is compressed and encrypted; data and metadata are displayed in an open XML form. Reading and writing data does not require any specific tools.

**Format description** The QDF format is described in the Snapform Schema. The Schema is found at [http://www.snapform.com/open\\_format/snapform.xsd](http://www.snapform.com/open_format/snapform.xsd). It is also listed in section 6.9.

The following code is an example which has been reduced to the essential elements.

```
<?xml version="1.0" encoding="UTF-8"?>
<QDFA version="1.5" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:noNamespaceSchemaLocation="http:/
/www.snapform.com/open_format/snapform.xsd"
xsi:rdp="http://www.snapform.com/open_format/rdp.xsd">
  <Metadata>
    <rdp:RDP xmlns:rdp="http://www.w3.org/1999/02/22-
rdp-syntax-ns#">
```

```

        <rdf:Description rdf:about="Dublin Core"
xmlns:dc="http://purl.org/dc/elements/1.1/">
        <dc:Title>REQUEST FOR SERVICES</dc:Title>
        <dc:Date>08-2005</dc:Date>
        <dc:Identifier>NRC FORM 160</dc:Identifier>
        </rdf:Description>
        </rdf:RDF>
    </Metadata>
    <Data>
        <snapform>
        <id>AF14A833-274C-C57A-0237-015E0D8A3F54</id>
        <values>
        <value name="contractor" type="4">>false</value>
        <value name="date1" type="5"></value>
        <value name="form_no" type="3"></value>
        <value name="hq" type="4">>false</value>
        <value name="licensee" type="4">>false</value>
        </values>
        </snapform>
    </Data>
    <Form><![CDATA[QlpoOTFBWSZTWREBGEgAA07////////////////
////////////////
+N71AAAAASKCgAAoFUAqm9fW4vVbs+3p5eOAAKAAjxMqp0dAo
...

DHALAPr8xrPEfcjiG+IZwcklbOMHyAwdKMAd10UGhvD2o4B60dUN6wwV1
AlsBg5EccH/4u5IpwoSAiAjqQA==]]>
    </Form>
</QDFA>

```

The elements are commented below:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The XML header.

```

<QDFA version="1.5" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://
/www.snapform.com/open_format/snapform.xsd"
xsi:rdp="http://www.snapform.com/open_format/rdp.xsd">

```

**QDFA** is the main element of a QDF/A form.

```
<Metadata>
```

The **Metadata** element contains the metadata entered in the Form settings window, **Metadata** tab (see Abschnitt 4.8.5).

```

        <rdf:RDF xmlns:rdp="http://www.w3.org/1999/02/22-
rdp-syntax-ns#">
        <rdf:Description rdf:about="Dublin Core"
xmlns:dc="http://purl.org/dc/elements/1.1/">

```

Metadata is used according to the Dublin Core specification.

```

        <dc:Title>REQUEST FOR SERVICES</dc:Title>
        <dc:Date>08-2005</dc:Date>
        <dc:Identifier>NRC FORM 160</dc:Identifier>
      </rdf:Description>
    </rdf:RDF>
  </Metadata>

```

```
<Data>
```

The **Data** block contains the data payload.

```
<snapform>
```

The **snapform** element indicates it is really Snapform forms data.

```
<id>AF14A833-274C-C57A-0237-015E0D8A3F54</id>
```

The **id** element is the form ID (see Abschnitt 4.8.2).

```
<values>
```

The **values** element contains the actual field values (element **value**).

```

        <value name="contractor" type="4">>false</value>
        <value name="date1" type="5"></value>
        <value name="date2" type="5"></value>
        <value name="form_no" type="3"></value>
        <value name="licensee" type="4">>false</value>
        <value name="mailstop" type="3"></value>
      </values>
    </snapform>
  </Data>
  <Form><![CDATA[Q1poOTFBWSZTWREBGEgAA07////////////////
////////////////
+N71AAAAASKCgAAoFUAqm9fW4vVbs+3p5eOAAKAAjxMqp0dAo

...

DHALAPr8xrPEfcjiG+IZwcklbOMHyAwdKMAd10UGhvD2o4B60dUN6wwV1
A1sBg5EccH/4u5IpwoSAiAjCQA==]]>
</Form>

```

The **Form** element contains a "blob" with the encoded form definition, which is interpreted by the Snapform Designer and the Snapform Viewer.

```
</QDFA>
```



## 6.3 Form Template, compressed (QDF/S)

### File extension

**.qdft**

### Explanation

Compact Snapform Template format. The whole content is compressed and encrypted. This format requires special tools to read the data including metadata.

The Template format forces the Snapform Viewer to perform a **Save as...** with a different file name.

### Format description

Pure binary format. Data can be read only via API (for a format description, see the documentation for the API).

## 6.4 Form Tempate, open (QDF/A)

### File extension

**.qdft**

### Explanation

Open Snapform Template format. Only the form description is compressed and encrypted; data and metadata are displayed in an open XML form. Reading and writing data does not require any specific tools.

The Template format forces the Snapform Viewer to perform a **Save as...** with a different name.

### Format description

The format description is identical with the description of the "standard" QDF/A format (see section 6.2).

## 6.5 Background layer

### File extension

**.hdt**

### Explanation

File format of the Background layer which has been created from a PDF document using the Snapform PDF Converter.

When opening a Field layer file in the Snapform Designer, a new document is created.

### Format description

Pure binary format. Data can be read only via API (for a format description, see the documentation for the API).

## 6.6 Field layer

### File extension

**.hdo**

### Explanation

File format of the Field layer which can be exported from the Snapform Designer.

When opening a Field layer file in the Snapform Designer, a new document is created.

### Format description

Pure binary format. Data can be read only via API (for a format description, see the documentation for the API).

## 6.7 Data, binary

### File extension

**.qdd**

### Explanation

Format of the binary data export and import file used by the Snapform Viewer.

This format is compressed and encrypted and is extremely compact. It requires specific tools to read and write data.

## Format description

Pure binary format. Data can be read only via API (for a format description, see the documentation for the API).

# 6.8 Data, open

## File extension

.xml

## Explanation

Format of the open data export and import file used by the Snapform Viewer.

This format is open and can be treated like any normal XML file. To read and write data, standard tools can be used.

## Format description

This format is a subset of the QDF/A format, and uses only the Data element of the Snapform Schema (see section 6.2).

# 6.9 The Snapform Schema

The Snapform Schema is published at [http://www.snapform.com/open\\_format/snapform.xsd](http://www.snapform.com/open_format/snapform.xsd).

The Schema is displayed below. It is recommended that you download the file and open it in an appropriate XML editor (XMLSpy, oXygen). There will be no further discussion of the Schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 4 U (http://www.xmlspy.com)
by Ringler Informatik AG -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="id"/>
  <xs:element name="value">
    <xs:complexType mixed="true">
      <xs:sequence minOccurs="0">
        <xs:element ref="rows"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

        <xs:attribute name="name" type="xs:string"
use="required"/>
        <xs:attribute name="type" type="xs:integer"
use="required">
        <xs:annotation>
        <xs:documentation>
Possible values for attribute 'type' are:
1 - Integer value
2 - Double value
3 - String value
4 - Boolean value
5 - Date value
6 - Table
7 - Array of bytes
8 - String value with index (used in dropdown
lists)
9 - Label value
10 - Negative double value</xs:documentation>
</xs:annotation>
</xs:attribute>
        <xs:attribute name="calculated" type="xs:boolean"
use="optional" default="false"/>
        <xs:attribute name="visible" type="xs:boolean"
use="optional" default="true"/>
        <xs:attribute name="printable" type="xs:boolean"
use="optional" default="true"/>
        <xs:attribute name="readonly" type="xs:boolean"
use="optional" default="false"/>
        </xs:complexType>
</xs:element>
<xs:element name="row">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="value" maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="rows">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="row" maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="values">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="value" maxOccurs="unbounded"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="snapform">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="id"/>
        <xs:element ref="values"/>
        </xs:sequence>
        </xs:complexType>
</xs:element>
<xs:element name="Keywords">

```

```

        <xs:complexType mixed="true">
        <xs:sequence>
        <xs:any processContents="skip"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="URL" type="xs:string"
use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Metadata">
        <xs:complexType>
        <xs:sequence>
        <xs:any processContents="skip"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="URL" type="xs:string"
use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Data">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="snapform"/>
        </xs:sequence>
        <xs:attribute name="URL" type="xs:string"
use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Form">
        <xs:complexType mixed="true">
        <xs:sequence>
        <xs:any processContents="skip"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="URL" type="xs:string"
use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Attachment">
        <xs:complexType mixed="true">
        <xs:sequence>
        <xs:any processContents="skip"
maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="Name" type="xs:string"
use="required"/>
        <xs:attribute name="Timestamp" type="xs:string"
use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Attachments">
        <xs:complexType>
        <xs:sequence>
        <xs:element ref="Attachment"/>
        </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="QDFA">
        <xs:complexType>
        <xs:all>

```

```
        <xs:element ref="Keywords" minOccurs="0"/>
        <xs:element ref="Metadata" minOccurs="0"/>
        <xs:element ref="Data" minOccurs="0"/>
        <xs:element ref="Form" minOccurs="0"/>
        <xs:element ref="Attachments" minOccurs="0"/>
    </xs:all>
    <xs:attribute name="version" type="xs:string"
use="optional"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

# 7

## API and Toolkits

An API as well as Toolkits are available. Their documentation is outside of the scope of this manual.

Further information is available from Ringler Informatik AG.