

# VisiWinNET 2005

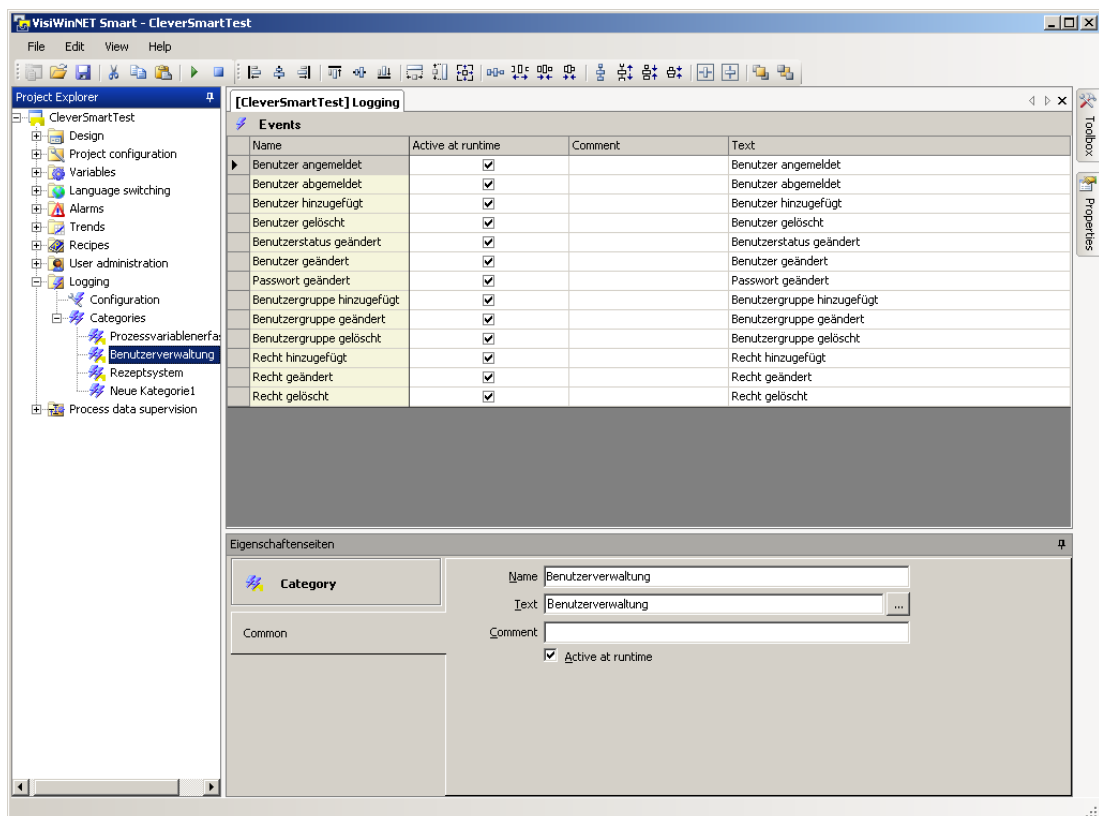
## Logging system



- **VisiWin**
- **VisiWinNET 2005**
- Common
- Class Library
- **Systems**
- Tools
- Technical Informations
- **Inosoft OPCServer**
- Basics and helping tools
- Protocols

# VisiWinNET 2005

## Logging system









The contents of this manual must not otherwise be used without explicit written consent from INOSOFT GmbH.

We have checked the contents of this manual for compliance with the described software. Discrepancies can, however, not be ruled out. For this reason we cannot guarantee full compliance. The contents of the manual are subject to regular checking for necessary updates/amendments. Such amendments will be made in the subsequent edition.

Suggestions for improvement are welcome.

## Legend

In order to point out particular paragraphs the following symbols are used in the INOSOFT documentations:

	<b>Attention</b>	Passages with this sign should be read – and observed – with particular attention.
	<b>Hint</b>	Important paragraph “additional information”
	<b>Tip</b>	Many roads lead to Rome; here a shortcut is to be found.
	<b>In work</b>	Functions that are in preparation or already implemented but not yet prepared for documentation.
	<b>Example execute</b>	Instructions to be carried out in an example
	<b>Observe result</b>	Results to be observed with carrying out the exemplary instructions

© / ™ / ®

Windows®, Windows NT®, Windows 2000®, Windows XP® are registered trademarks of the Microsoft company.

Further product names marked ® are trademarks of the appropriate manufacturers.

INOSOFT GmbH created on

VisiWinNET Version: from 6.04.000

created on 08.06.2010

# Contents

<b>1 Preamble</b> .....	<b>1</b>
<b>2 Introduction to the VisiWinNET logging system</b> .....	<b>2</b>
<b>3 Design</b> .....	<b>4</b>
3.1 Logging Implementation .....	5
<b>4 Definitions of the Logging system</b> .....	<b>7</b>
4.1 Definitions and Product versions of the logging system.....	7
4.2 Category .....	7
4.2.1 Edit Categories.....	8
4.3 Logging event.....	9
4.3.1 Edit Logging Events.....	10
4.4 Logging parameter .....	10
<b>5 Parameters of the definitions of the logging system</b> .....	<b>12</b>
5.1 All parameters of the logging definitions .....	12
5.2 Parameters in alphabetic sequence .....	12
5.2.1 Active at runtime.....	12
5.2.2 Comment.....	13
5.2.3 Index .....	13
5.2.4 Logging text.....	14
5.2.5 Name .....	14
5.2.6 Source.....	14
5.2.7 Text .....	15
5.2.8 Type.....	16
<b>6 Log files</b> .....	<b>17</b>
6.1 System categories and events .....	18
6.2 Texts of the logging events.....	19
<b>7 Logging server configuration</b> .....	<b>21</b>
7.1.1 Standard runtime .....	21
7.1.2 Compact runtime.....	24

# 1 Preamble

## About this manual

This manual contains specific information on the VisiWinNET logging system, among others the explanation of the involved components, the surface settings, and the description of the data format of the logging files.

## Questions and Problems

For technical questions and problems please contact your responsible INOSOFT agent or the INOSOFT GmbH Support under +49 (5221) 16 66 02 or email: [Support@INOSOFT.com](mailto:Support@INOSOFT.com)

Frequent questions and problems are dealt with on our homepage under [www.inosoft.com](http://www.inosoft.com)

There you will also find a support area for direct contact with our Main Office.

## 2 Introduction to the VisiWinNET logging system

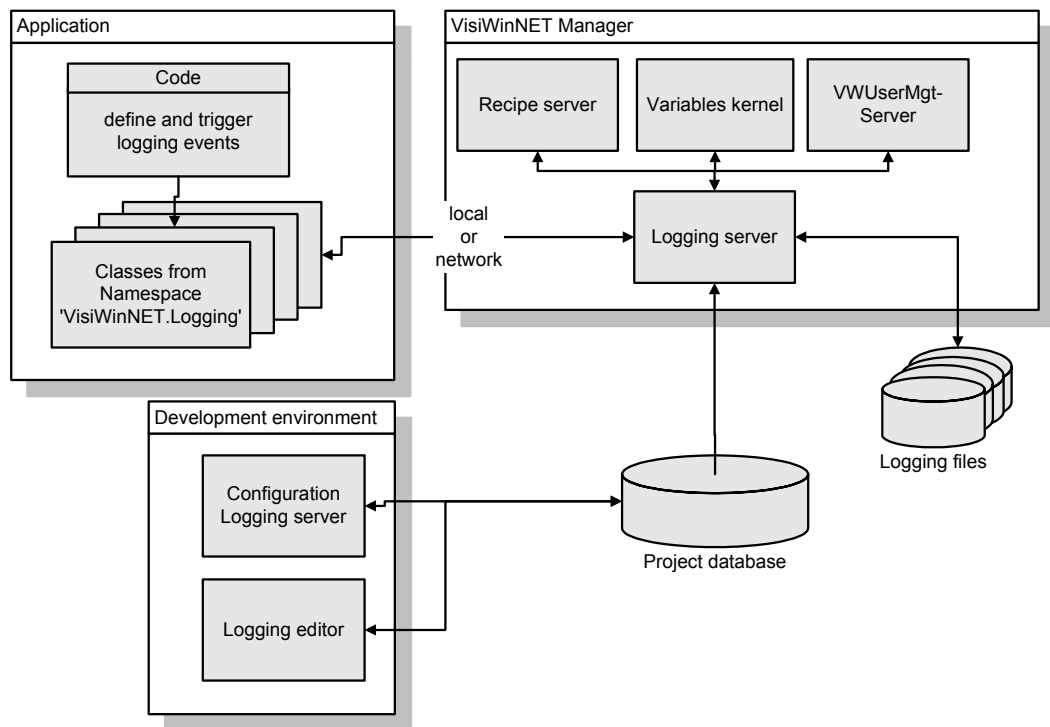
The VisiWinNET logging server serves to record events in logging files. A firm set of events is provided by the system. Further events can be freely defined in the application.

The events provided by the system refer to login/logoff procedures in the user administration as well as to variable changes in the variable kernel and the recipe server.

Application-specific logging events are defined in the logging editor at development time. At runtime they are triggered via the interfaces of the 'VisiWinNET.Logging' namespace. Every event definition can be named unambiguously, and allocated to a category. Beside the text information user names, notes etc. can be filed in the occurring events.

The following block scheme shows the components of the logging system. Further information about the components is indicated as cross-references.

VisiWinNET logging system components



### Logging server

During the initializing phase the logging server logs on with the server components to be monitored. The project database supplies the logging server with the necessary information for administrating the logging files and the logging events to be evaluated. Subsequently relevant events are picked up by the logging server in the server components to be monitored, and written into the appropriate current logging file. New event definitions from the visualization applications are written into the the structure information of the logging files. If subsequently a self-defined logging event is triggered by the application it can be incorporated into the data area of the logging files.


### Recipe server/ Variable kernel/

Variable kernel, user administration and recipe system are automatically monitored by the logging system. The server components to be monitored supply the data for the appropriate

<b>VWUserMgtServer</b>	entries into the logging files.
<b>Classes in the 'VisiWinNET.Logging' / 'VisiWinNET.LogBook' *) namespace</b>	<p>The "LogManager" class implemented in this namespace serves to define and trigger self-defined logging events. The "HistoricalLogging" class allows reading access to the data of the logging files. Via the "HistoricalLoggingNote" class additional notes can be added to the logging data. For further information see "VisiWinNET Class Library" manual.</p> <p>*) In Compact the 'VisiWinNET.LogBook' namespace is used.</p>
<b>Code</b>	Triggering of own logging events must be implemented by the projector himself.
<b>Project database</b>	The project database contains information for the logging server. Relevant logging settings can be made via the configuration dialog of the logging server. The automatic logging of variable value changes can be individually switched on or off for every variable in the process connection editor.
<b>Logging editor</b>	At development time own categories and logging events are defined here.
<b>Configuration logging server</b>	The configuration allows settings for the logging file administration. Further information can be found in the "Configuration of the protol server" chapter in this manual.
<b>Logging files</b>	The logging files are Access 2000 databases. Logging files are saved with the file ending „vwl“. Further information about the buildup of the logging files can be found in the "Logging files" chapter" in this manual.

## 3 Design

VisiWin provides an editor for the projection of the logging. This is integrated with the development environment with the installation of VisiWinNET. The projected definitions are stored in the project database.

The logging editor is represented in the VisiWinNET Project Explorer by the  symbol. This is the main access node for the definitions in the logging.

After the first click on the "Logging" node the editor initializes itself in the Project Explorer. The following access nodes are added:

**Configuration Logging** Contains as a VisiWinNET properties page global settings that control the logging performance.

**Categories** Contains the logging categories to be projected.

Beneath the "Categories" node system categories are already added with a new project:

**Process variable recording** System category containing the logging events of the variable kernel. These events allow the recording of variable value changes as well as the start/stop of applications.

**User administration** System category containing the logging events of the user administration. These events allow the monitoring of user logons/logoffs and the user administration.

**Recipe system** System category monitoring the recipe definitions for changes.

Self-defined categories are displayed parallelly with these nodes. The appropriate logging events are displayed in the table editor.

### 3.1 Logging Implementation

This chapter lists in note form some typical logging problems under VisiWinNET. The key question is: "Where do I find what?". This, therefore, is an outline of the projection steps.

Task	Projection
<b>Recording variable value changes</b>	The process variable definitions contain the parameter "Record value in the log server". If this parameter is activated an appropriate entry is written into the log files when the value of the variable changes.
<b>Determining location of log file(s)</b>	In the configuration of the logging system (node "Logging→Configuration") in the project explorer there are settings for the administration of the log files. Here it is determined in which directory and under which name the log files are to be created.
<b>Defining and triggering further logging events</b>	<p>The definition of own logging events mostly relates to processes from the visualization application. When was which interaction carried out by the operator? What comments or additional information were lodged on this?</p> <p>The definition of these events is realized in the logging editor by creating new categories, logging events and logging parameters. Triggering, however, must be implemented in the application through functions of the control elements or source code.</p> <ul style="list-style-type: none"> <li>• Functions of the control elements: In the 'Events' property a control elements event can be linked with the system function "Logging→Log".</li> <li>• Source code: The static function 'LogManager.Log' in the 'VisiWinNET.Logging' namespace ('VisiWinNET.LogBook' in Compact) triggers the appropriate logging event.</li> </ul>

### Watching the recorded data



- The 'HistoricalLoggingList' control element lists the recorded events from the log files.
- Program access to the data of the log files is possible through the 'GetHistoricalData' method of the 'HistoricalLogging' class.

Control element as well as class contain a filter.

Future Release?!?: When this manual was printed it was not yet known whether the above mentioned functions will all be implemented in version 6.2.

## 4 Definitions of the Logging system

### 4.1 Definitions and Product versions of the logging system

Below a collection of the definitions of the VisiWinNET logging system. In addition the information is given as to in which product version the definition is contained.

Definition	VisiWinNET Standard	VisiWinNET Compact	VisiWinNET Embedded
<b>Category</b>	✓	✓	✓
<b>Logging event</b>	✓	✓	✓
<b>Logging parameter</b>		✓	✓

### 4.2 Category

Categories serve to divide logging events into different origin areas.

VisiWinNET itself uses categories by allocating the system events (firmly predetermined set of available logging events) to the appropriate system categories. Presently, the following systems cooperate with the logging server:

- Recipes
- Data access
- User administration.

The appropriate categories are hence predetermined.

With the help of categories the projector can develop an own hierarchy of logging events.

Amongst others, categories offer the following additional functions:

- Specification of a localizable name that can be displayed instead of the definition name at runtime.
- Activation/deactivation at development as well as runtime.

#### Parameters of the category definition

Name	Description
<b>Active at runtime</b>	Is definition to be interpreted as an active definition by the runtime system.
<b>Comment</b>	Freely choosable comment on definition.
<b>Name</b>	Unequivocal designator of definition.
<b>Text</b>	Localizable name

### 4.2.1 Edit Categories

Categories are displayed in the Project Explorer under the "Logging" node. Every category contains:

- an editable set of parameters: Here the name and a localizable text can be set.
- logging events where applicable: After highlighting a category the appropriate event definitions are displayed in the table editor.

**Create new category** Through the "New" entry in the context menu of the "Logging" node.

**Edit parameters** A click on a category loads the appropriate parameters to the VisiWinNET properties page. Here the parameter values can be edited.

**Delete** Through the "Delete" entry in the context menu of a category node the definition is deleted. In the process the contained events are deleted, too.



The "Process Variable Recording", "User Administration" and "Recipe System" system categories cannot be deleted but deactivated through the "Definition active" parameter.

### 4.3 Logging event

Logging events describe the real content of the log files. Through the definition of a logging event the data to be recorded are determined. Key components of the logging event are the log text and the appropriate information (e.g. variable values).

Logging events from within system categories (recipes, process variable gathering, user administration) are allocated to certain events that occur in the appropriate systems. The triggering of these events is hard-wired in the appropriate systems which means that these events are recorded automatically without involving the developer.

Self-projected logging events must be triggered from within the application. The access is through specification of the name.

Amongst others logging events offer the following functionalities:

- Localizable text to name the source of the event.
- Subdivision into categories for source area definition.
- Activation/deactivation at development as well as runtime.
- Linking of process variable values, index texts or freely defined parameters from within the application (only in Compact).

#### Parameters of the logging events

Name	Description
<b>Active at runtime</b>	Is definition to be interpreted as an active definition by the runtime system.
<b>Comment</b>	Freely choosable comment on definition.
<b>Logging text</b>	Localizable text on the logging event (for Compact only).
<b>Name</b>	Unequivocal designator of definition.
<b>Text</b>	Localizable name.

### 4.3.1 Edit Logging Events

Logging events are projected in categories, and displayed in the table editor. Every event contains an editable set of parameters: Here, amongst other details, it is determined by which name the event can be triggered, and which text appears in the logging databases.

The table editor of the logging is opened through a click on an appropriate node in the Project Explorer.

The editor offers the following functions for the projection of logging events:

- |                         |   |
|-------------------------|---|
| <b>Create new event</b> | Through the "New" entry in the context menu of the table editor a new event is added to the project.  |
| <b>Edit parameters</b>  | The VisiWinNET properties page displays the parameters of the event highlighted in the table editor. Editing parameters is, however, also allowed directly in the fields of the table editor.   |
| <b>Delete</b>           | One or multiple events can be deleted by: <ul style="list-style-type: none"><li>• first highlighting the definitions to be deleted (click on the selector column at the l.h. table margin, where applicable with the Ctrl or Shift key held down for multiple selection)</li><li>• then selecting the "Delete" entry in the context menu of the table editor.</li></ul> |

## 4.4 Logging parameter

Logging parameters

- Process variable values
- Localizable user-defined texts
- Localizable names for variables
- Values from within the application that are transferred by the program when the logging event is triggered.

Selection of the information type is determined in the "Type" parameter.

In the parameters of a logging event definition the logging parameters are made available as insertible components in "Logging text".

**Parameters of the logging parameters**

<b>Comment</b>	Freely selectable comment on the definition
<b>Index</b>	Identifier for the sequence of the logging parameters in a logging event
<b>Name</b>	Unequivocal name of the logging parameter
<b>Source</b>	Closer definition of the additional information (see "Type")
<b>Text</b>	Localizable identifier for the parameter
<b>Type</b>	Selection of the information type (see "Source")

## 5 Parameters of the definitions of the logging system

### 5.1 All parameters of the logging definitions

The description of the parameters contains the following information:

Block	Description
<b>Parameter for</b>	Lists the definitions containing this parameter.
<b>Description</b>	Provides a description of the parameter functionality.
<b>Database field</b>	The name of the index column in the VisiWinNET project datenbase.
<b>Data type</b>	The data type of the parameter.
<b>Standard value</b>	The value that is allocated as a standard for the parameter after a new definition has been added.
<b>max. length</b>	The maximum length of possible entries.

### 5.2 Parameters in alphabetic sequence

Below an alphabetic parameter collection. In addition the information as to by which VisiWinNET product version the parameter is supported.

Parameter name	VisiWinNET Standard	VisiWinNET Compact	VisiWinNET Embedded
<b>Active at runtime</b>	✓	✓	✓
<b>Comment</b>	✓	✓	✓
<b>Index</b>		✓	✓
<b>Name</b>	✓	✓	✓
<b>Source</b>		✓	✓
<b>Text</b>	✓	✓	✓
<b>Type</b>		✓	✓

#### 5.2.1 Active at runtime

Parameter for	Category, Logging event
Description	The "Active at runtime" parameter deactivates the definition at development time. The activation at runtime can be effected via the "Logging" class.

### 5.2.2 Comment

Parameter for  
Description

Category, Logging event, Logging parameter

A field for a freely choosable comment is available for each definition. Comments serve as a help at development time. They have no function at runtime.

### 5.2.3 Index

Parameter for  
Description

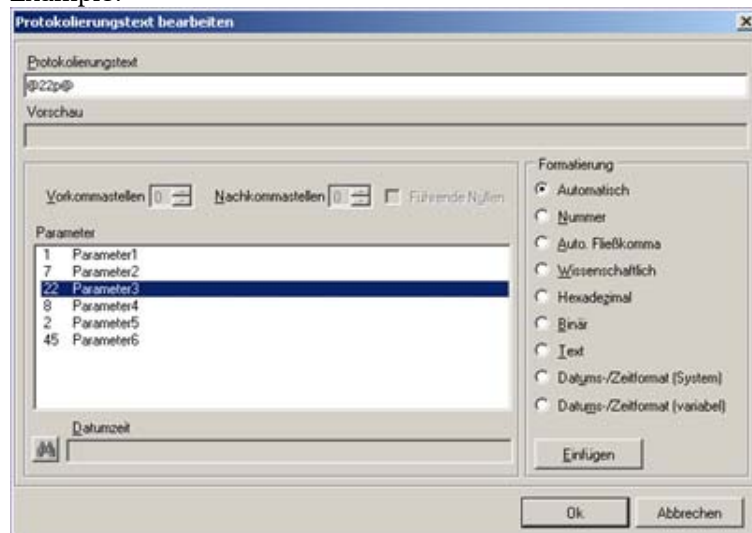
Logging parameter

Identifier for the sequence of the logging parameters in a logging event.

The index is used:

- For passing user-defined logging parameter values in the "VisiWinNET.LogBook.LogManager.Log" function. Here a field of parameter values is expected in the sequence determined in the index.
- For adding to the logging text of the logging event. If a logging parameter is added here the specified index is expected as identifier.

Example:



"Parameter 3" added with the index "22" is added as text component "@22p" to the logging text.

## 5.2.4 Logging text

Parameter for



Description

Logging event

Currently the logging text is only contained in the Compact version. In the Standard runtime the "Text" parameter is to be used instead.


Determines the text of the logging event to be recorded. The logging text may contain synonyms for logging parameters. These synonyms are enclosed in "@" characters. They contain a parameter index and a formatting command.

Example:

**"Machine started with @1m3.2@ revolutions"**

Contains the formatting command: **"@1m3.2€"**

- 1 being the logging parameter with the index 1
- m3.2 being the numeric indication with leading zeroes, three positions before, and two positions after the decimal point.

The developer is supported with the projection of the logging text through a dialog. This dialog is opened through the  button to the right of the appropriate input field on the VisiWinNET properties page.

## 5.2.5 Name

Parameter for

Description

Category, Logging event, Logging parameter

The unequivocal designator of the definition in the project database.

The name is typically used by the logging server as an identification characteristic of the definition.

The name is expected as a parameter in the classes of the 'VisiWinNET.Logging' namespace. These objects communicate with the logging server, in the process passing on the accessed function with the definition name (to which the function is to be applied).

## 5.2.6 Source

Parameter for

Description

Logging parameter

In the logging parameter the source determines the name of the definition from which an information is to be read.

The available definition names and the information to be read depend on the setting of the "Type" parameter.

## 5.2.7 Text

Parameter for

Category, Logging event, Logging parameter

Description

Via the text a localizable text to the definition can be specified. If at runtime an evaluation of the recorded data is to be displayed it makes sense in a project with implemented localization to compose and display the texts of the logging events and categories, too, in the different languages.

Via the language selection list in the editor the language can be selected in which the runtime texts are to be projected.

In the index text editor display of all texts in all languages is possible. An index text group is created for every projected category under the "Localization→Logging" node. After navigating to one of these nodes the appropriate index texts can be edited in the index editor of the text index editor.

If no texts were allocated the definition name is shown instead at runtime.



In the Standard runtime texts from within the localization are allocated to the system logging events. The allocation is describes in the chapter "Texts of the logging events".

### 5.2.8 Type

Parameter for

Beschreibung

Logging parameter

Determines which information type the logging parameter contains. The available information types determine from where the value of the parameter is to be read.

If the parameter value relates to a process variable or a localizable text the appropriate definition name is expected in the "Source" parameter.

Setting

Description

#### **Variable value**

The logging parameter value is at runtime read from a process variable. The name of the process variable is specified through the "Source" parameter.

#### **Variable text**

The logging parameter value is at runtime read as localizable text from a process variable. The name of the process variable is specified through the "Source" parameter.

#### **Index text**

The logging parameter value is at runtime read from a text of the localization. The name of the process variable is specified through the "Source" parameter.

#### **User defined**

The logging parameter value is at runtime passed by the program in the access of the "Log" function (VisiWinNET.LogBook.LogManager or in an 'Events' function of the "Log" system function).

Here the "Source" parameter has no function.

## 6 Log files

The log files are MSAccess-Databases. Four tables, including event definitions as well as the event files, are located in these databases.

### Table "Categories"

Column	Description
<b>CatID</b>	Unique category identification.
<b>Name</b>	Category designator.
<b>TextID</b>	Numerical index text reference in the project database for language switchable text output.

### Table "Events"

Column	Description
<b>Event</b>	Unique logging event identification within the category.
<b>CatID</b>	Reference to the field "CatID" in the table "Categories" referring to the category.
<b>Name</b>	Logging event designator
<b>TextID</b>	Numerical index text reference in the project database for language switchable text output.

### Table "Logbook"

Column	Description
<b>ID</b>	Unique data record identification (assigned automatically).
<b>CatID</b>	Reference to the field "CatID" in the table "Categories" referring to the category.
<b>Event</b>	Reference to the field "Event" in the table "Events" referring to the logging event definitions.
<b>User</b>	User name, logged in during event triggering.
<b>Date</b>	Exact date and time of the logging event.
<b>Text</b>	Logging event text.
<b>TextID</b>	Numerical index text reference in the project database for language switchable text output.
<b>HelpFile</b>	Optional help file name.
<b>HelpID</b>	Optional help page number (HelpContextID)
<b>HasNotes</b>	Information about log entry note existence.
<b>SortBuffer</b>	

**Table "Notes"**

Column	Description
ID	Unique data record identification (assigned automatically).
ParentID	Reference to the field "ID" in the table "LogBook" referring to a recorded event.
Date	Exact date and time of notes.
User	User name, logged on.
Note	Note information.

**6.1 System categories and events**

The following categories and events are being used by the system:

Category	CatID	EventID	Event
<b>VWE-Kernel (Data recording)</b>	<b>20</b>	<b>1</b>	System start
	<b>20</b>	<b>2</b>	System stop
	<b>20</b>	<b>3</b>	Variable change
<b>User administration</b>	<b>60</b>	<b>1</b>	User logged on
	<b>60</b>	<b>2</b>	User logged off
	<b>60</b>	<b>3</b>	New User
	<b>60</b>	<b>4</b>	User deleted
	<b>60</b>	<b>5</b>	User state altered
	<b>60</b>	<b>6</b>	User altered
	<b>60</b>	<b>7</b>	Password changed
	<b>60</b>	<b>8</b>	New User class
	<b>60</b>	<b>9</b>	User class altered
	<b>60</b>	<b>10</b>	User class deleted
	<b>60</b>	<b>11</b>	New right class
	<b>60</b>	<b>12</b>	Right class altered
	<b>60</b>	<b>13</b>	Right class deleted



<b>Password changed</b>	USERMGT_TRACE_18
<b>New user group</b>	USERMGT_TRACE_19; USERMGT_TRACE_19A
<b>User group changed</b>	USERMGT_TRACE_20 : USERMGT_TRACE_20A
<b>User group deleted</b>	USERMGT_TRACE_21:
<b>New right</b>	USERMGT_TRACE_22; USERMGT_TRACE_22A
<b>Right changed</b>	USERMGT_TRACE_23; USERMGT_TRACE_23A
<b>Right deleted</b>	USERMGT_TRACE_24

### "Recipes" category

The texts for these events are found in the text group:

"Components→ServerComponents→Recipe"

Event	Index text
<b>Recipe value changed</b>	RECIPE_TRACE_34
<b>Recipe read from PLC</b>	RECIPE_TRACE_20
<b>Recipe sent to PLC</b>	RECIPE_TRACE_21
<b>Recipe loaded</b>	RECIPE_TRACE_22
<b>Recipe saved</b>	RECIPE_TRACE_23

## 7 Logging server configuration

The configuration settings of the logging are loaded to the VisiWinNET properties page by highlighting the "Logging→Configuration" Project Explorer node.


The settings differ between the Standard and Compact runtime systems.

### 7.1.1 Standard runtime

The logging server configuration allows the following settings:

#### Tab page "History"

##### File path

Path specification for all logging files. The path declaration can be specified absolute (the button  opens a dialog for directory selection) as well as relative to the project directory (the password "<ProjectDir>" initializes the relative path specification i.e.: "<ProjectDir>\Alarme") .


##### File name

Specifies a file name. The file name is either specified automatically or manually.

##### automatically

The file name consists of the current date (where appropriate also the current time) at which the file was created.

##### manually

Similar to a language-related text the file name in the index text editor can contain a dynamically linked process variable value. The  button opens a dialog to link a dynamic process variable value.

##### Event changing file

Through the file changeover event the moment of new file creation is determined. Possible settings:

Daily: With change of date a new message history file is created.

Weekly: With change of week a new file is created.

Monthly: With change of month a new file is created.

Yearly: With change of year a new file is created.

Trigger: With trigger variable setting a new file is created.

Never / via interface: The system does not carry out a file change on its own. There is, however, the possibility to trigger a file change from within the program code through the functions in the "VisiWinNET.Logging" namespace.

##### Trigger variable

Specifies a digital variable name, which serves as trigger for the file changeover. The field is active only if in the "File changeover event" the setting "Trigger" was processed.

**File mode**

Determines, how the file is to be saved.

Fast: The log file will remain opened until the swapping-out event. This will enhance the file access performance. With a crash though, correct file content cannot be guaranteed.

Secure: if this option is active, the logging server will close the log file after each write access. With a system crash this will intensify safety, but can possibly slow down the system.

**Max. file count**

The parameter "Maximum number of files" defines the maximum number of log files that can be created on the hard disk. With exceeding the specified number, first saved files will be deleted. This serves to avoid hard disk overflow.

Setting this value to "0" (default) means that no limitation will be executed.

**Max. file size**

The parameter "Maximum file size" defines the maximum memory size on the hard disk that can be taken up by the log files. With exceeding the specified (in kilobyte) size here, first saved files will be deleted. This serves to avoid hard disk overflow.

If the parameter "File mode" is set to "quick", it can come to overshooting of the here prescribed size. Size comparison will be proceeded only after data, located in the internal storage was recorded in the file. A large internal buffered data set possibly leads to the necessity of first saved files deletion to keep the maximum total file size.

Setting this value to "0" (default) means that no limitation will be executed.

**Save user name**

This option defines if the current user name is to be saved during the recording of a log entry.

**Text for 'no user logged on'**



Determines the text that is saved as a user name if no user is logged on in the application. The parameter is activated when the "save user name" option is activated.

**Tab page "file format"**

Field	Description
<b>File format</b>	Determines the format in which the logging databases are to be saved. A choice can be made between Microsoft Access® 2000 and MSSQL server databases.
<b>SQL Server name</b>	If the SQL server is selected under file format the computer name of the computer on which the SQL server was installed, and on which the logging databases shall be created can be specified here.
<b>Use Windows NT integrated.../ Use specific...</b>	The choice of the setting available here depends on the releases in the database. The integrated security uses the role of the logged on user in the network to authenticate. Input of user name and password must be administrated in the database..
<b>User name</b>	Name of user for authentication with the database .
<b>Password</b>	Password of user for authentication with the database .

### 7.1.2 Compact runtime

The configuration of the logging server allows the following settings for the Compact runtime:

Field	Description
<b>File path</b>	Specifies the path for all historical log files. The path specification can be absolute (the  button opens a dialog to select a directory) as well as relative to the project directory (the "#ProjectDir#" keyword stands at the beginning of the relative path specification, e.g.: "#ProjectDir#\Logging").
<b>File name</b>	Specifies a file name. The file name is either specified automatically or manually. <p><b>automatically</b></p> <p>The file name consists of the current date (where appropriate also the current time) at which the file was created.</p> <p><b>manually</b></p> <p>Similar to a language-related text the file name in the index text editor can contain a dynamically linked process variable value. The  button opens a dialog to link a dynamic process variable value.</p>
<b>File change event</b>	Through the file change event it is determined when a new file is to be created. The following settings are possible: <p>daily: a new alarm history file is created with the date change</p> <p>weekly: a new file is created with the week change</p> <p>monthly: a new file is created with the month change</p> <p>annually: a new file is created with the year change</p> <p>Trigger: a new file is created when the trigger variable is set.</p> <p>Never / via interface: The system does not carry out a file change on its own. There is, however, the possibility to trigger a file change from within the program code through the functions in the "VisiWinNET.Logging" namespace.</p>
<b>Trigger variable</b>	Specifies the name of a digital variable that serves as a trigger for a file change. The field is only active if the "file change event" is set to "Trigger".
<b>Cache derivate time</b>	Specifies the maximum time by which saving the alarm information is delayed. This is a protective measure for media with restricted writing cycles (e.g. "Flash" cards with 100,000 guaranteed writing cycles). By this measure the lifespan of the card can be improved by collecting logging information in the RAM first. That information is then written to the Flash card in one collective operation.

**Maximum cache size** Specifies the maximum number of logging entries to be cached in the RAM before being written to the data medium.

This, too, is a protective measure for media with restricted writing cycles.

**Maximum number of files** The "Maximum number of files" parameter determines the maximum number of historical log files that can be created on the data medium. If the number set here is exceeded the oldest files are deleted. This is to save the hard disc from overflow.

Setting this value to "0" (default) means that no limitation will be executed.

**Maximum total file size** The "Maximum total file size" parameter determines the maximum room that the historical log files can assume on the data medium. If that size (specified in kilobyte) is exceeded the oldest data are deleted. This is to save the hard disc from overflow.

Setting this value to "0" (default) means that no limitation will be executed.