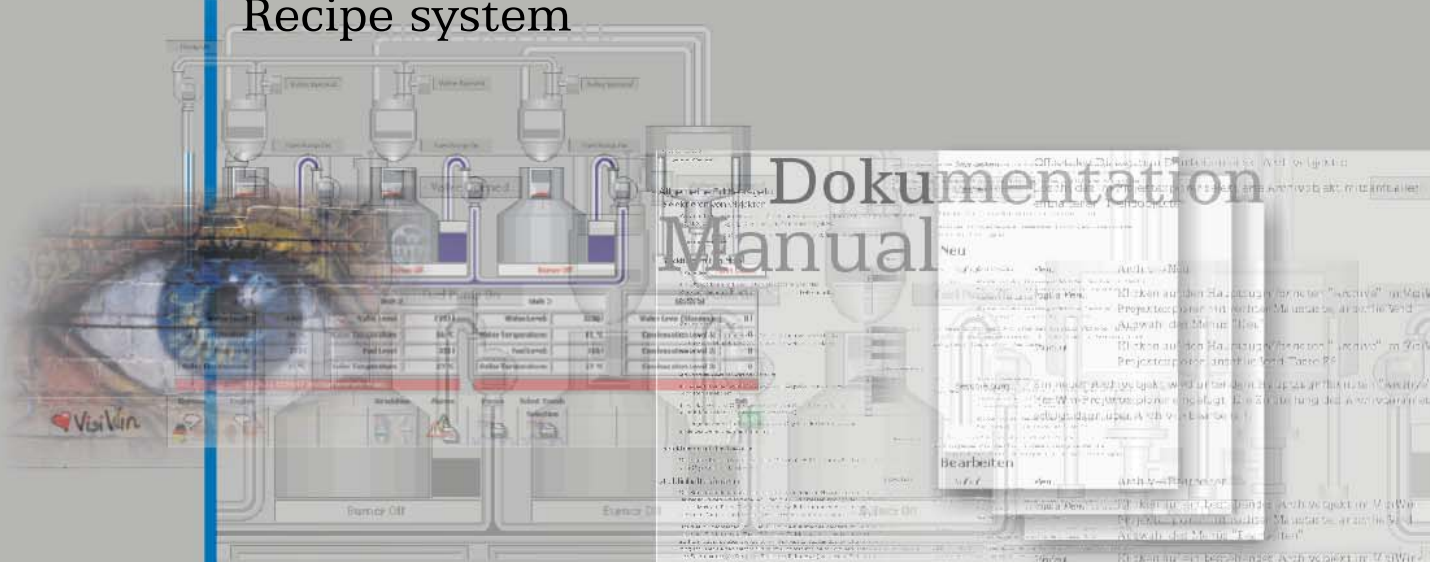


VisiWinNET 2005

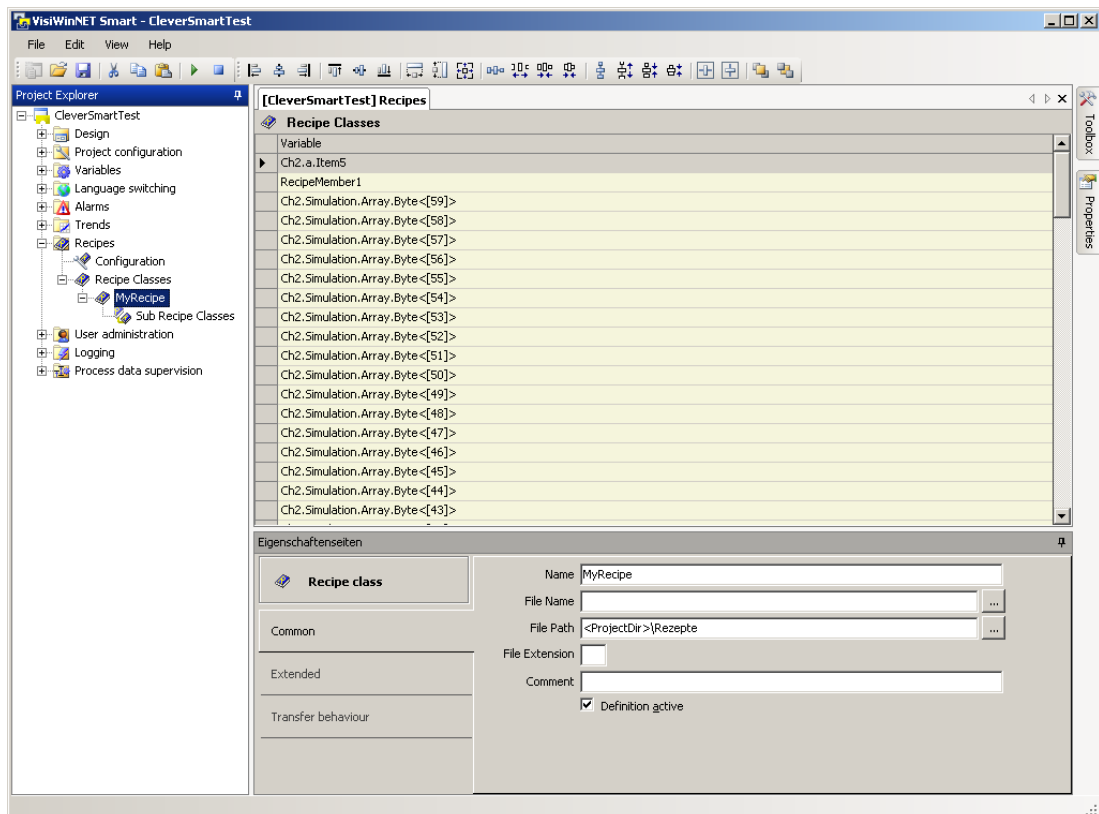
Recipe system



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

VisiWinNET 2005

Recipe 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:

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

1 Preamble	1
2 VisiWinNET Recipe System	2
3 Recipe editor	5
4 Recipe System definitions	6
4.1 Definitions and product versions in the recipe system.....	6
4.2 Recipe.....	6
4.2.1 Edit Recipe	7
4.3 Recipe element	8
4.3.1 Edit Recipe element.....	8
5 Recipe definition parameters	9
5.1 All recipe system definition parameters	9
5.2 Parameter in alphabetic order	9
5.2.1 Activate Change Logging	10
5.2.2 Comment.....	10
5.2.3 Definition active	10
5.2.4 Description variable.....	10
5.2.5 File extension.....	11
5.2.6 File Name variable.....	11
5.2.7 Include Sub recipe class values	11
5.2.8 Load last recipe file on startup	12
5.2.9 Load trigger	12
5.2.10 Max. file count.....	12
5.2.11 Name.....	12
5.2.12 Read trigger	13
5.2.13 Recipe file path.....	13
5.2.14 Recipe values absolute/relative.....	13
5.2.15 Save changed values directly.....	13
5.2.16 Save changed variable kernel values direct.....	14
5.2.17 Save trigger	14
5.2.18 Scheduled factor variable	14
5.2.19 Standard factor	14
5.2.20 Sub recipe.....	15
5.2.21 Variable	16
5.2.22 Write changed values direct	17
5.2.23 Write trigger.....	17
6 Recipe server configuration	18
6.1 Settings.....	18

1 Preamble

About this manual

This manual contains specific information on the VisiWinNET recipe system, among others the explanation of the involved components, the operating reference of the editor, and the description of the recipe system definitions.

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

Frequent questions and problems are dealt with on our homepage under www.inosoft.com

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

2 VisiWinNET Recipe System

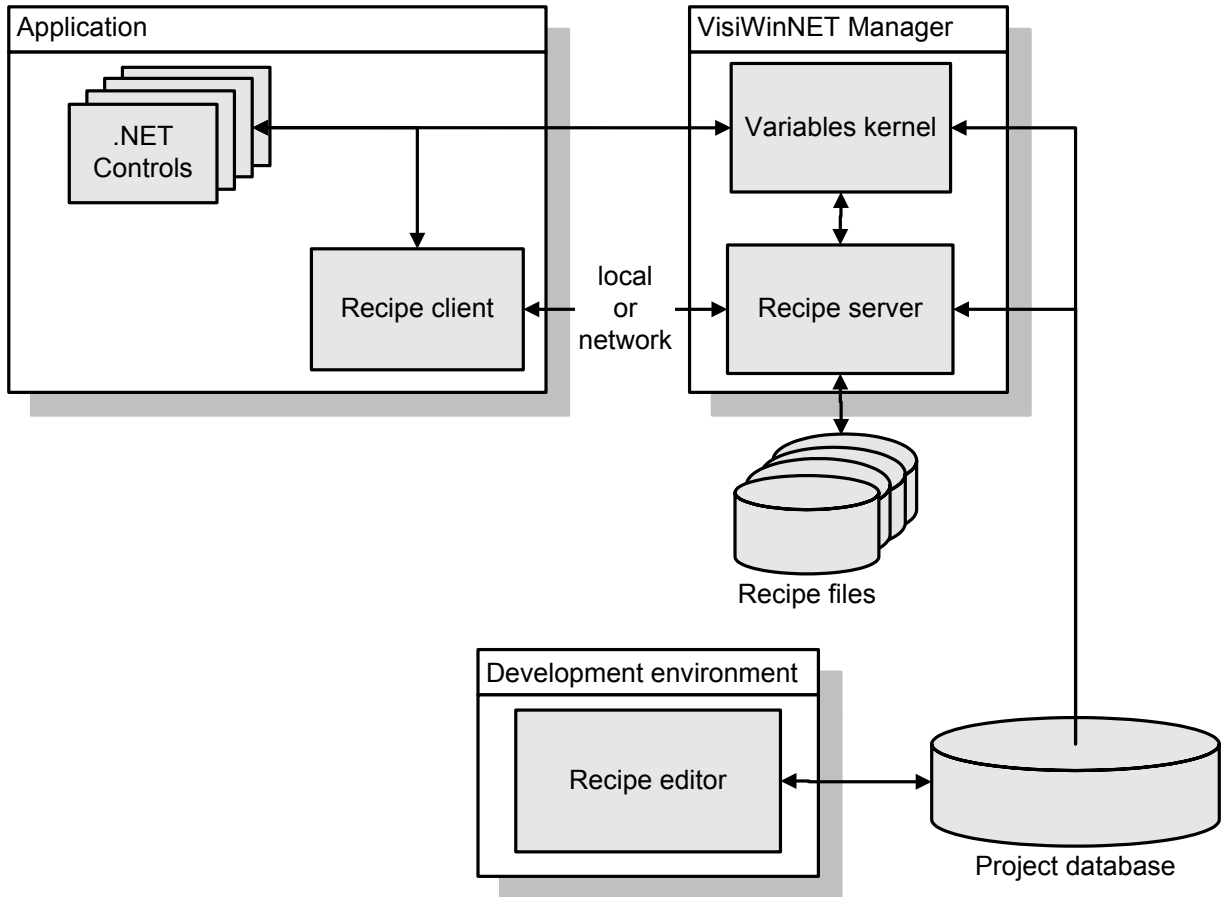
Visualization recipes specify process value sets, which usually define the machine settings for product manufacture from a product range. Over recipe writing in the PLC, switching between the specific products being manufactured is enabled.

The VisiWinNET Recipe System meets all requirements of a comfortable Recipe management:

- Through designing of recipe definitions, the process variables essentially required for recipe processing are selected from the Process Connection.
- The recipe data transfer between PLC and visualization is both read and write controllable. Over the writing access, existing recipes are transferred to the PLC. The reading access enables the transfer of machine-optimised data records to the visualization system. Transfer control is possible from the visualization as well as from the PLC.
- The recipe data transfer between visualization system and recipe files is both read and write controllable. Through the reading access, recipe file data is loaded in the visualization system. The writing access enables permanent availability of recipe data, optimised on the machine. Manual controlling from the visualization application as well as automatic controlling over the PLC is made possible here.

The recipe system components will be introduced on base of the following block circuit diagram. Continuing information on the components is provided as cross- reference.

VisiWinNET Recipe system components



Component	Description
Variable kernel	The Variables Kernel provides the process values involved in the recipes. Further information is provided in the handbook "Process connection"
Recipe server	<p>The recipe server buffers the recipe files centrally. It serves as interface between recipe client, variable kernel and recipe files. Apart from the recipe data transfer to the connected recipe clients (and thus to the control elements in the visualization) it supports four transfer functions essentially:</p> <ul style="list-style-type: none"> • Read: The recipe values will be transferred from the Kernel into the recipe server buffer. • Write: The recipe values will be transferred from the internal buffer of the recipe server into the Kernel. From there they will be forwarded to the PLC.

- **Load:** The recipe data stored in the buffer will be read from a recipe file. File control (Name, directory and file description) will be processed through the parameter sets included in the recipe definition.
- **Save:** Recipe data stored in the buffer will be written into a recipe file. File control (Name, directory and file description) will be processed through the parameter sets included in the recipe definition.

Information on the setting possibilities of the recipe server is provided in the chapter "Recipe server configuration" (Chap.: 6) in this handbook.

Recipe client

As central gathering point in a visualization application, the recipe client enables access to the recipe data buffered on the recipe server and the transfer functions in the recipe server.

Control elements

The control elements in the recipe mode can be switched between two views. They can present either current values from the variable kernel or else buffered values from the recipe server.

Project database

The project database contains the recipe system definitions. During development time the process variables involved in a recipe are to be defined (Recipe elements). Through the recipe definition parameter sets it is defined

- where and how recipe data is saved and
- which process variable can control the transfer function in the recipe server with PLC-control.
- which standard factors are to be used with relative recipe size specification and
- how sub-recipes are to be included.

Recipe editor

The recipe editor is the design tool for the recipe system definitions. It is displayed in the development environment in the project explorer. Hints for operating of the recipe editor are provided in the chapter of the same name in this handbook.


Recipe files

Contain the recipe value lists, combined on the basis of the recipe definitions. Information on the recipe file format is provided in the chapter "Recipe server configuration" (Chap.: 6) in this handbook.

3 Recipe editor

The following chapter contains the operating reference of the recipe editor. With the editor definitions can be designed in the project database. During run time, the definitions are used for data exchange control between Kernel and Client-Visualisation applications by the recipe server.

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

The recipe editor is represented in the VisiWinNET Project Explorer by the  symbol.

After the first click on the "Recipes" node the editor initializes itself: All recipe definitions already defined in the project database are added as subnodes to the "Recipe classes" node beneath.

In addition the "Configuration Recipe System" node is added. Here system-wide settings such as the format of the recipe files are made.

The table editor is opened through the "Display Editor" context menu or a click on a recipe system node in the Project Explorer. The table editor displays the contents of the selected node.

"Recipe Classes" node	All recipe definitions of the project are displayed in the table editor.
<Recipe Definition> node	All elements projected in the recipe are displayed in the table editor.

4 Recipe System definitions

4.1 Definitions and product versions in the recipe system

Below a collection of the definitions of the VisiWinNET recipe system. In addition it is shown which product version contains the definition.

Definition	VisiWinNET Standard	VisiWinNET Compact	VisiWinNET Embedded
Recipe	✓	✓	✓
Recipe element	✓	✓	✓

4.2 Recipe

Recipe definitions are presented under the appropriate access node in the project explorer. They contain a parameter set, which is stored in the project database and provide the information for the recipe server, how the recipe definitions must be interpreted during run-time.

Recipe definitions provide among other things following functionality:

- Indication of digital process variables, which trigger the four basic functions "Read", "Write", "Load" and "Save".
- Relative or absolute interpretation of recipe values by scheduled- and standard factor.
- Recipe nesting over sub recipes.



Applies to VisiWinNET Smart users: Depending on the selected target device there are some settings that are not available or only configurable with limitations. For more information, see the Device Catalog.

Recipe definition parameters

Name	Description
Activate Change Logging	Changed values will be saved in the recipe file.
Comment	Optional object comment.
Definition active	The definition is to be interpreted as valid definition by the run-time system.
Description variable	Process variable, which provides a descriptive text for the recipe.
File extension	Extension of all recipe files, created by the recipe definition.
File name variable	Determines the process variable that contains the recipe file name.
Include Sub recipe class values	Determines, whether recipe values from sub recipes will simultaneously be stored in the file.
Load last recipe file on startup	Determines whether the recipe file loaded latest in the previous application session is to be immediately loaded into the appropriate process variables when the application is started.

Load trigger	Digital process variable, which value change, leads to value load from the recipe file.
Max. file count	Determines how many recipe files may be created for this definition.
Name	Unique designator of the definition in the recipe system.
Read trigger	Digital process variable, which value change, leads to recipe value read from the Kernel.
Recipe file path	Determines the path, under which the recipe files are stored.
Relative/absolute recipe values	Determines, whether recipe value randomizing will be processed by standard- and scheduled factor.
Save changed values directly	Automatic save in the recipe file without trigger function.
Save changed variable kernel values directly	Automatic save in the recipe file without trigger function.
Save trigger	Digital process variable, which value change, leads to value Save in the recipe file.
Scheduled factor variable	Process variable, which contains the recipe server data multiplier with the relative declaration of recipe values.
Standard factor	Recipe file data multiplier with the relative recipe value declaration.
Write changed values directly	Automatic writing into the PLC without trigger function.
Write trigger	Digital process variable, which value change, leads to value writing into the Kernel.

4.2.1 Edit Recipe

Recipes are displayed in the Project Explorer under the node "Recipes→Recipe Classes". Every recipe definition contains:

- an editable set of parameters: Here the name or for example the allocation of file names and descriptions can be set.
- recipe elements where applicable: Recipe elements are displayed in the table editor. They contain the information as to which process values are contained in the recipe files.

The editor provides the following functions for the projection of recipes:

Create new recipe	Through the "New" entry in the context menu of the "Recipe Classes" node.
Edit parameters	A click on a recipe node loads the appropriate parameters to the VisiWinNET properties page. Here the parameter values of the definition can be edited.
Delete	Through the "Delete" entry in the context menu of a recipe node the definition is deleted. In the process all secondary definitions (recipe elements) are also deleted from the project.

4.3 Recipe element

Recipe elements determine, which process variables belong to a recipe. The linkage can be processed both directly by the designation of the process variable and indirectly via the linkage to a sub recipe. Sub Recipe elements can be stored conditional on the recipe definition settings either separately in a recipe file or together with the superior recipe definition data.

Recipe element parameters

Name	Description
File name variable	Determines the process variable that contains the recipe file name in which the sub recipe data will be stored.
Sub recipe	Recipe insertion as Recipe element.
Variable	Process variable, which value is to be included in the recipe.

4.3.1 Edit Recipe element

Recipe elements are projected in recipes, and displayed in the table editor. Every recipe element is distinguished by a variable designator. By this, process variables determined as recipe elements are linked with the functions of the recipe server.

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

The editor offers the following functions for the projection of recipe elements:

Create new recipe element	Through the "New" entry in the context menu of the table editor a dialog is displayed that offers all process variables of the project for selection.
Delete	One or multiple recipe elements can be deleted by: <ul style="list-style-type: none"> • 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) • then selecting the "Delete" entry in the context menu of the table editor.

5 Recipe definition parameters

5.1 All recipe system definition parameters

The parameter description contains following information:

Block	Description
Parameter for	Lists the definitions, which contain this parameter.
Description	Indicates a parameter functionality description.
Database box	Table column name in the VisiWinNET-Project database.
Data type	Parameter data type.
Default value	Standardized value, which is assigned to the parameter after new object insertion.
Max. length	Max. length of possible input.

5.2 Parameter in alphabetic order

Below an alphabetic collection of the parameters. In addition the information is provided as to which VisiWinNET product version supports the parameter.

Parameter name	VisiWinNET Standard	VisiWinNET Compact	VisiWinNET Embedded
Activate Change Logging	✓	✓	✓
Comment	✓	✓	✓
Definition active	✓	✓	✓
Description variable	✓	✓	✓
File extension	✓	✓	✓
File name variable	✓	✓	✓
Include Sub recipe class values	✓	✓	✓
Load last recipe file on startup	✓		
Load trigger	✓	✓	✓
Max. file count		✓	✓
Name	✓	✓	✓
Read trigger	✓	✓	✓
Recipe file path	✓	✓	✓
Relative/absolute recipe values	✓	✓	✓
Save changed values directly	✓	✓	✓
Save changed variables kernel values directly	✓	✓	✓

Save trigger	✓	✓	✓
Scheduled factor variable	✓	✓	✓
Standard factor	✓	✓	✓
Sub recipe	✓	✓	✓
Variable	✓	✓	✓
Write changed values directly	✓	✓	✓
Write trigger	✓	✓	✓

5.2.1 Activate Change Logging

Parameter for	Recipe
Description	Setting of the parameter "Activate change history" causes that not only the current recipe values will be saved in the file. But the first re-written values will survive as well and can be read again as required through the interfaces of the Recipe client.

5.2.2 Comment

Parameter for	Recipe
Description	For each recipe definition an optional comment box is provided. Comments serve as developer support during development time. They have no operability during run time.

5.2.3 Definition active

Parameter for	Recipe
Description	Determines, if the definition validity is recognized by the runtime system. The parameter deselecting is equivalent with the comment out.

5.2.4 Description variable

Parameter for	Recipe
Description	Indicates the process variable (of data type VT_BSTR), which value contains a descriptive text to the recipe. When storing the recipe file this value is stored as description. While loading the description is transferred from the recipe file to the process variable.

5.2.5 File extension

Parameter for

Recipe

Description

Suffix of all recipe files that were created through the recipe definition. Saving new recipe files an "R" will be put in front of the indicated suffix automatically. With load filtering in the file selection view is possible through the suffix.

5.2.6 File Name variable

Parameter for

Recipe, Recipe element

Description

Determines, which process variable of data type VT_BSTR contains the recipe file name. Under this name the recipe will be stored/loaded.



The content of the "Variable for file name" is only used if the trigger functions (Load trigger, Save trigger of the recipe definition are triggered. If a 'RecipeClassHandler' or 'RecipeClass' object is used the "Variable for file name" has no function.

Sub recipe data can be stored either as characteristic file or in the parent recipe. If the parameter "Include Sub recipe class values" is activated, then, the parameter "File name variable" is without meaning.

5.2.7 Include Sub recipe class values

Parameter for

Recipe

Description

Determines, whether the sub recipe values will be stored in the parent file.

Settings Description

False Recipe values are stored in the sub recipe file. On this occasion, the parameter "file name" indicates the name of the file.

True Values are stored in the parent recipe file. No operability of the parameter "file name".

5.2.8 Load last recipe file on startup

Parameter for	Recipe
Description	<p>Determines whether the recipe file loaded latest in the previous application session is to be immediatedly loaded into the appropriate process variables when the application is started.</p> <p>Following the start of the application the recipe cache of a recipe class is usually empty. Through the option described here the recipe cache can be initialized with the latest loaded recipe file. Immediately after the values are automatically transmitted to the appropriate process variables and, with this, to the PLC.</p> <p>Between application runs the system remembers the name of the latest opened recipe file in the project databank.</p>

5.2.9 Load trigger

Parameter for	Recipe
Description	<p>Determines a process variable, which value change to "1" triggers value load from a recipe file into the recipe server. After loading the process variable value is automatically reset to "0". The process variable can assume either a Boolean or an integer data type. Over the bit number definition in the variable selection dialog, the trigger bit will be selected.</p> <p>The name of the file to be loaded is expected in a process variable that is to be specified in the „File name variable" parameter.</p>

5.2.10 Max. file count

Parameter for	Recipe
Description	<p>Determines how many files may be created for this recipe definition. This setting is a protective mechanism for systems with little memory capacity.</p> <p>When saving a recipe file the system checkst he number of existing files by the file extension and the file path. If the maximum number of files is exceeded the system returns the "ExceededMaxFiles" error in the "SaveDone" event.</p>

5.2.11 Name

Parameter for	Recipe
Description	The unique definition designator in the project database.

5.2.12 Read trigger

Parameter for

Recipe

Description


Determines a process variable, which value change to "1" triggers the value read from the Kernel into the recipe server. After loading the process variable value is automatically reset to "0". The process variable can assume either a Boolean or an integer data type. Over bit number definition in the variable selection dialog, the trigger bit will be selected.

5.2.13 Recipe file path

Parameter for

Recipe

Description

Determines the path, under which the recipe files will be stored/loaded. The path declaration can be both absolute (The button  opens the dialog window for directory selection) and relative to the project directory (The keyword "<ProjectDir>" starts the relative path declaration, i.e.: "<ProjectDir>\recipes").

5.2.14 Recipe values absolute/relative

Parameter for

Recipe

Description

Recipe values can be interpreted absolute and relative. With absolute interpretation values are exchanged directly with the Kernel. Relative interpretation enables randomizing in reference to a "standard set" through standard factor and scheduled factor variable.

5.2.15 Save changed values directly

Parameter for

Recipe

Description

The parameter "Save changed values directly" causes that through the application (i.e. in the VisiWinNET-Control elements) changed values will be written in the recipe files directly through the recipe server (therefore without releasing of the trigger function "Save").



In the process the changed process values are written into the latest used recipe file. In order for the values to be stored the "Load" or "Save" function must have been executed previously at least once.

5.2.16 Save changed variable kernel values direct

Parameter for	Recipe
Description	The parameter "Save changed variable kernel values directly" causes that values, changed in the variable kernel will be written in the recipe files directly through the recipe server (therefore without releasing of the trigger function "Save").

5.2.17 Save trigger

Parameter for	Recipe
Description	Determines a process variable, of which value change to "1" triggers value Save from the recipe server into the recipe files. After loading the process variable value is automatically reset to "0". The process variable can assume both, a Boolean and an integer data type. Over bit number determination in the variable selection dialog the trigger bit will be selected. The name of the file to be loaded is expected in a process variable that is to be specified in the "File name variable" parameter.

5.2.18 Scheduled factor variable

Parameter for	Recipe
Description	Recipe values can be interpreted absolute and relative. With absolute interpretation, values are exchanged directly with the Kernel. Relative interpretation enables randomizing in reference to a "standard set" through standard factor and scheduled factor variable. Values are randomized according to following formula: Value VWOPC-Kernel = $\frac{\text{Standard factor} \times \text{Value Recipe server}}{\text{Value Scheduled factor variable}}$

5.2.19 Standard factor

Parameter for	Recipe
Description	Recipe values can be interpreted absolute and relative. With absolute interpretation, value exchange directly with the Kernel is processed. Relative interpretation enables randomizing in reference to a "standard set" through standard factor and scheduled factor variable. Values are randomized according to following formula: Value VWOPC-Kernel = $\frac{\text{Standard factor} \times \text{Value Recipe server}}{\text{Value Scheduled factor variable}}$

5.2.20 Sub recipe

Parameter for

Recipe element

Description

Indicates the name of a recipe definition, which is to be merged as sub recipe.

The function of sub-recipes

Sub-recipes always make sense if a recipe can be broken down into modules. The combination of the individual modules through a "main recipe" (a recipe containing the individual modules as sub-recipes) will result in ever new production processes. An example (from a cookery book) would be the production of different cakes that can be broken down into the "case" and "topping" modules. Through combining these modules all sorts of cakes from a strawberry gateau through a blueberry cake to a chocolate rusk can be produced. In this case the modules (or sub-recipes) contain

for the topping:

- strawberry
- blueberry
- chocolate

for the case:

- baked pastry case
- shortcrust
- rusk.

By combining the individual sub-recipes even the (admittedly exotic) strawberry rusk can be produced. The main recipe to be loaded for this purpose contains the names of the appropriate sub-recipe files. These sub-recipe files in turn contain the real machine parameters of the individual modules.

Changes within a sub-recipe go straight into all main recipes that use that sub-recipe. On one hand this makes data maintenance considerably easier. On the other hand, it needs to be observed that certain measures must be taken in the case of a possible obligation to produce supporting documentation (FDA). In such a case changes to a sub-recipe must not enter the production process unreported so that it is documented that a main recipe is used that involves a changed sub-recipe.

The transfer performance of sub-recipes

The transfer of the sub-recipe values between process, recipe cache and recipe files is arranged so that sub-recipe values cannot by error be overwritten by the transfer functions of the main recipe.

- Loading: by loading a main recipe all sub-recipe elements are automatically loaded into the recipe cache.
- Saving: the saving process only writes the main recipe values into the recipe file. To save sub-recipes the appropriate "Save" method of the sub-recipe must be accessed.
- Reading: with the reading process only the main recipe elements are transferred from the variable kernel into the recipe cache.
- Writing: writing includes the main recipe as well as the sub-recipe elements.
- Editing mode: switching of the control elements between variable kernel and recipe cache through the main recipe applies to the elements of the main recipe only.

This transfer performance means that in the application the appropriate administration of the recipe files (creation of new files, editing of variable values in the recipe cache, ...) must be carried out for every individual sub-recipe. Usually an own form is projected for every sub-recipe in which the values can be edited, and through which the transfer functions for the sub-recipe values can be accessed.

The administration of the main recipe includes all elements, i.e. on top of the recipe variables also the file names of the sub-recipe files to be considered.

An example for the design of an application form containing the values as well as the transfer control elements can be found in the "VWRecipeDemo" project example.

5.2.21 Variable

Parameter for

Recipe element

Description

Defines a process variable, which value is to be included in the recipe. Through determination of a valid process variable designator, the appropriate value is adopted into the recipe definition functions.

5.2.22 Write changed values direct

Parameter for	Recipe
Description	The parameter "Write changed values directly" causes that through the application (i.e. in the VisiWinNET-Control elements) changed values will be forwarded to the PLC through the variable kernel directly (therefore without releasing of the trigger function "Write").

5.2.23 Write trigger

Parameter for	Recipe
Description	Determines a process variable, which value change to "1" triggers value writing from the recipe server into the Kernel. After loading the process variable value will automatically be reset to "0". The process variable can assume both a Boolean and an integer data type. Over bit number determination in the variable selection dialog the trigger bit will be selected.

6 Recipe server configuration

6.1 Settings

The configuration of the recipe server allows system-wide settings.

The configuration dialog of the recipe system is accessed through the equally named Project Explorer node. The settings are displayed on the VisiWinNET properties page.

Setting	Description
Max. concurrent orders	Defines the number of jobs that the recipe server can buffer. There, a job is a transfer function request (load, save, read or else write /of a recipe). Transfer function requests will be buffered and processed in succession.
XML/MS-Access	Defines the recipe files format. It can be chosen between XML (Extended markup language)-files and Access-databases.
Ok	Confirms the entries for the project file and closes the dialog.
Cancel	Drops the entries and closes the dialog.