PMS MICADO

Virtual Product Modeling System (VP/MS)

Getting Started

For VP/MS component versions:

- **Vframe German**
- **Workbench English V2.10a**
- **Designer English V3.25 build 021**
- **IFOS Report Editor German Test 1.57b**
- **IFOS Report Viewer German V1.9**

Copyright and trademarks

Copyright

PMS MICADO VP/MS
VP/MS Getting Started Manual, December 1999

For more information about VP/MS, please contact:

PMS MICADO SoftwareConsult GmbH
Reutherstr. 1a-c
53773 Hennef / Germany
Tel. : (+49) (0) 22 42 - 871 - 400
FAX : (+49) (0) 22 42 - 871 - 455
Email: info.micado@notes.compuserve.com
Web: www.pmsmicado.com

Trademarks

Windows and *Windows/NT* are registered trademarks of the Microsoft Corp.

Table of Contents

Documentation Overview

VP/MS Getting Started (this manual)

Intended audience

The intended audience for this manual includes anyone who wants to quickly gain hands-on experience with VP/MS. This manual provides a very detailed step-by-step VP/MS tutorial that allows anyone with no previous VP/MS experience to quickly master basic VP/MS tasks and thus gain a thorough understanding of VP/MS concepts.

What this manual contains

This manual contains the following sections:

- **'Copyright and trademarks' (page 3)**.
- **'Table of Contents' (page 5)**.
- **'Documentation Overview' (page 13)**. This section.
- **'Installing VP/MS' (page 15)**. Explains how to install VP/MS components.
- **'Tutorial' (page 17).** A step-by-step tour of the basic concepts and techniques of VP/MS.
- **'A. Tools, Directories, Files, File Types' (page 187).** Describes the following:
  - Tools (Workbench, etc.)
  - Directories (C:\VPMS, etc.)
  - Files (vpms.dll, etc.)
  - File types (.pms, etc.).
- **'B. Glossary' (page 193).** Includes any special terms used throughout this manual.
- **'List Of Figures' (page 199)**.
- **'Index' (page 205)**.

Recommendations for completing the step-by-step tutorial

For those with no VP/MS experience

It is recommended that you complete the entire tutorial. By completing the tutorial you become familiar with the VP/MS toolset and how to use this toolset to complete common tasks (creating a model, creating a lay-out, printing, etc.). The tutorials in the user guides for the tools (Workbench, Designer, VFrame, IFOS) assume that the reader has already mastered the material presented in the VP/MS tutorial.

For those with VP/MS experience

Browse the VP/MS tutorial contents for topics that are new to you. The tutorial referenced user guides for the tools (Workbench, Designer, VFrame, IFOS) which provide more advanced information for the given topic.

Conventions used in this manual

The following conventions are used in this manual.

- A term being used for the first time is bold and italic. For example: ***VP/MS***.
- Dialog names and menu entries are displayed in bold, with a forward slash between nested entries. For example: **Model / Test**.
- Code is fixed-width Courier. For example:

  ```
  COMPUTE
  ```

Abbreviations used in this manual

- **W**: Workbench.
- **WT**: Workbench test.
- **D**: Designer.
- **DT**: Designer test.
- **RE**: IFOS Report Editor.
- **VF**: VFrame.
- **DBW32**: Dbwin32 debugger.

Reader comments

Any comments you have concerning this manual can be sent to:

Other sources of information

From the CD ROM

Manuals

The following PMS Micado manuals are referenced throughout this manual. They are available on the VP/MS CD ROM in directory **\Docs** and are copied to the **\VPMS\Docs** directory on your harddisk during installation:

- **vframe.pdf**: VFrame User's Guide.
- **workbench.pdf**: Workench User's Guide. Includes:
  - Instructions for creating realistic Workbench insurance models (the tutorial uses the simplest models possible to demonstrate Workbench functionality).
  - Workbench reserved properties.
  - Workbench functions and operators.
  - Contents of the consult.ini and xplconf.ini files.
- **designer.pdf**: Designer User's Guide. Includes:
  - Designer elements and properties.
- **ifos_reditor.pdf**: IFOS Report Editor User's Guide. Includes:
  - IFOS command language.
  - IFOS print language.
  - IFOS internal variables.
- **ifos_rviewer.pdf**: IFOS Report Viewer User's Guide.

Help

Context-sensitive on-line help is available by simply clicking the **Help** button or **F1** for major dialogs. General help is also available via the **Help** in the main menu.

Training from PMS Micado

This manual is provides complete information to help you start using VP/MS as soon as possible. However, it is also recommended to consider enrolling in a training course from PMS Micado that is customized to your specific needs.

For more information about training courses, please contact:

PMS MICADO SoftwareConsult GmbH
Reutherstr. 1a-c
53773 Hennef / Germany
Tel. : (+49) (0) 22 42 - 871 - 400
FAX : (+49) (0) 22 42 - 871 - 455
Email: info.micado@notes.compuserve.com
Web: www.pmsmicado.com

Installing VP/MS

Installation requirements

The following is required for installation:

- An **IBM PC compatible** with **Windows NT** or **Windows 95**.
- For **Windows NT**: **Administrator priviledges**. .
- PMS Micado **VP/MS CD ROM**.

Installing VFrame

On a computer WITHOUT a previous installed version of VFrame

1. Double click on **E:\Vframe\setup32.exe** ("E" = CD ROM drive letter). The **Willkommen** dialog appears.
2. Click **Weiter**. The **Installationsmethode wählen** dialog appears. The default is **Eigenständiger Arbeitsplatz**.
3. Click **Weiter**. The **Ziellaufwerk wählen** dialog appears. The default dialog is **C:**.
4. Click **Weiter**. The **Installationsbereit** dialog appears.
5. Click **Weiter**. The files are copied to your harddisk. The dialog **Die Installation ist abgeschlossen** appears.
6. Click **Weiter**.

On a computer WITH a previous installed version of VFrame

CAUTION: It is recommended not to deinstall the previous version of VFrame. However, if you decide to deinstall a previous VFrame version, then **create backups of the following files**:

- C:\VPMS\VFrame\menu**consult.ini**
- C:\VPMS\VFrame\menu**xplconf.ini**

The above files will be deleted during the deinstallation.

7. Double click on **E:\Vframe\setup32.exe** ("E" = CD ROM drive letter). The **Willkommen** dialog appears.
8. Click **Weiter**. The **Zielverzeichnis** dialog appears.
9. Click **Weiter**. A dialog appears for determining whether or not database files from the previous installation should be overwritten. The default selection is **nein, vorhandene Datenbank erhalten**.
10. Click **Weiter**. The **Installationsbereit** dialog appears.
11. Click **Weiter**. The files are copied to your harddisk. The dialog **Die Installation ist abgeschlossen** appears.
12. Click **Weiter**.

Installed files

VFrame is now installed on your computer and is available from:

- **Start Menu** entry **Programs / VFrame32 / VFrame**.
- Desktop icon **VFrame32**.

The executable is **C:\VPMS\Vframe\vframe32.exe**.

Installing Workbench

13. Double click on **E:\Workbench\Setup.exe** ("E" = CD ROM drive letter). The **Welcome** dialog appears.
14. Click **Next**. A dialog with information appears.
15. Click **Next**. The dialog **Select destination drive** appears. **C:** is the default drive. NOTE: It is highly recommended to use the default setting.
16. Click **Next**. The dialog **Ready to install!** appears.
17. Click **Next**. The files are copied to your harddisk. The dialog **Installation completed!** appears.
18. Click **Finish**.

Installed files

Workbench is now installed on your computer and available from:

- **Start Menu** entry **Programs / CAF VPMS / VPMS Workbench**.
- Desktop icon **VPMS Workbench**.

The executable is **C:\VPMS\Wbench\Vpms32.exe**.

Installing Designer / IFOS (Report Editor / Report Viewer)

On a computer WITHOUT a previous installed version of Designer

19. Double click on **E:\Designer\Setup.exe** ("E" = CD ROM drive letter). The **Welcome** dialog appears.
20. Click **Next**. The dialog **Select destination drive** (for Designer) appears. **C:** is the default drive/directory. NOTE: It is highly recommended to use the default setting.
21. Click **Next**. The **Backup replaced files?** dialog appears. **Yes** is selected.
22. Click **Next**. The files for Designer are copied to your harddisk. The dialog **Select Destination Directory** (for IFOS) appears. **C:\Programs\CAF\IFOS** is the default drive/directory.
23. Click **Next**. The files for IFOS are copied to your harddisk. The dialog **Installation Completed!** appears.
24. Click **Finish**.

On a computer WITH a previous installed version of Designer

25. Double click on **E:\Designer\Setup.exe** ("E" = CD ROM drive letter). The **Welcome** dialog appears.
26. Click **Next**. The dialog **Select destination drive** (for Designer) appears. **C:\VPMS\Designer** is the default drive/directory. NOTE: It is highly recommended to use the default setting.
27. Click **Next**. The **Backup replaced files?** dialog appears. **Yes** is selected.
28. Click **Next**. The files for Designer are copied to your harddisk. The dialog **Installation Completed!** appears.
29. Click **Finish**.

Installed files

Designer is now installed on your computer and available from:

- **Start Menu** entry **Programs / CAF VPMS / VPMS Designer**.
- Desktop icon **VPMS Designer**.

The executable is **C:\VPMS\Designer\vsd.EXE**.

IFOS Report Editor is now installed on your computer and available from:

- **Start Menu** entry **Programs / CAF Anwendungen / IFOS Report Editor**.

The executable is **C:\Programs\CAF\IFOS\Ifosre.exe**.

IFOS Report Viewer is now installed on your computer.

The executable is **C:\Programs\CAF\IFOS\Cafrg.exe**.

Entering the license code

A license code must be entered; otherwise, you will only be able to use a shareware version of the tools.

30. In file **C:\Winnt\vpms.ini**: Enter the following lines in the section **[VP/MS Workbench]**:

licensename=**{your license name}**

licensenr=**{your license number}**

Tutorial

1. Tutorial overview

"Example is the school of mankind, and they will learn at no other." Edmund Burke.

The tutorial provides step-by-step instructions that demonstrate how to use the VP/MS tools while introucing VPMS concepts. The following is an outline of the tutorial.

Using the examples; Specifying file extension defaults

The following is important before actually starting the tutorial.

- **'2. Using the included Zyx example files' (page 24)**. Demonstrates how to start the tutorial at any chapter using using the completed files for the previous chapter (included on the CD).
- **'3. Specify extension defaults for VPMS files (Windows NT)' (page 25)**. Demonstrates how to set the default program for files created with VPMS tools.

Part 1: Basics

In these chapters you will become familar with VP/MS basics.

VFrame menus (VF)

- **'4. VFrame menus (Notepad)' (page 27)**. VFrame menus are created with .ini files in the directory **C:\vpms\vframe\menu**. The .ini files are created during installation of VFrame.

Agents and customers (VF)

- **'5. Manage agents (VF)' (page 29)**. Agent information is stored in database **C:\vpl_apps\data_bas\anwender.dbf** (created during installation of VFrame). Access to this database (ie, creating and deleting agents) and user interface dialogs are provided by **vfkv32d.dll** (installed during the installation of VFrame). vfkv32d.dll functions are accessed from the VFrame main menu.
- **'6. Manage customers (VF)' (page 31)**. Customer information is stored in database **C:\vpl_apps\data_bas\person.dbf** (created during installation of VFrame). Access to this database (ie, creating and deleting customers, changing customers information) and user interface dialogs are provided by **vfkv32d.dll** (installed during the installation of VFrame). vfkv32d.dll functions are accessed from the VFrame main menu.

Consultation (W,D,VF)

- **'7. Create model (.pms) and runtime model (.vpm) (W)' (page 33)**. The model contains a complete description of the Products, Objects, Events, and Compensations and the relationships between them. A runtime model is required for use with the layout within VFrame.
- **'8. Create layout (.vpl) and runtime layout (.vpc) (D)' (page 37)**. The layout defines the agent (end-user) interface. A runtime layout is required for use in VFrame.
- **'9. Configure VFrame consultation Zyx (Notepad)' (page 41)**. The consultation menu for VFrame is specified in consult.ini. The location for the files for the runtime model and layout (created above) that are used for the VFrame consultation are specified in xplconf.ini.
- **'10. Open consultation; Create vorgang (VF)' (page 43)**. Information entered during a consultation (using the runtime layout in VFrame) is stored in database **C:\vpl_apps\data_bas\vorgang.dbf** (created during installation of VFrame). A vorgang contains all entered information since previous vorgang was created. When viewing a vorgang in VFrame, all changes in all previous vorgangs are taken into account.

Report (RE, W)

- **'11. Create report (.cat) (RE)' (page 46)**. Describes how to use the IFOS RE to create a .cat file that defines the structure of the printed report for the consultation.
- **'12. Create model attributes and tables for print support (W)' (page 48)**. Describes the special attributes and tables required in the model for print support.
- **'13. Open consultation; Generate report (VF, RV)' (page 49)**. Demonstrates how to print the report.

Customer information in the consultation/report (W,D,VF,RV)

The customer information can be added to the consulation (the layout) and thus also added to the report.

- **'14. Add customer information to the consultation (W,D)' (page 51)**. Create the model, layout, report, and the interface files required between the runtime layout and the databases.
- **'15. Add customer information to the report (RE,RV) ??' (page 55)**. Modify the report to include the customer information from the layout.
- **'16. Add push buttons to the layout for accessing vfkv32d.dll dialogs' (page 56)**. The changes for customer information made within the layout are saved as a vorgang. This is normally not desireable. Push buttons are added to the layout that allows to open the vfkv32d.dll dialogs from the layout.

Part 2: Versioning and testing

Basic testing for models, layouts, report, etc. was introduced in previous chapters. The following chapters explore testing in more detail and introduce versioning (a versioning mechanism is implemented for models within the Workbench; other types of files (layouts, reports, etc.) must be backed up manually).

Model versioning / testing (W,WT)

- **'17. Model versions (vpms.ini, .pms) (W)' (page 61)**. Describes the versioning process for models. This allows errors in the model to be found within the Workbench.
- **'18. Testing the model (.pms, .vpm) (W, WT)' (page 62)**. Describes how to use the Workbench Test tool to test the model within or outside the Workbench / before and after creating a runtime model. Also describes test cases and searching for errors.

DBWin32 debugger

- **'19. DBWin32 debugger messages (vpms.ini, W, DBWin32)' (page 65)**. Describes how to use the Debugger. Internal messages generated by VPMS tools are recorded in DBWin32.

Layout testing (D,DT)

- **'20. Testing the layout (.vpl, vpc) (D, DT)' (page 66)**. Allows errors in the layout to be found within the Designer (a Workbench runtime model is required).

Report testing / konzeptvorschau (IFOS, RE, RV)

- **'21. Testing the report with IFOS debug window (not available under NT ??)' (page 67)**. If an error occurs in IFOS, it is trapped and displayed by this debugger.
- **'22. Analyzing the report with Konzeptvorschau (RE, RV)' (page 68)**. While not actually a test or debugging tool, the Konzeptvorschau displays information that can help you find errors in the report.

VFrame testing (VF)

- **'23. Testing in VFrame (VF) ??' (page 69)**. There is no tool specifically for debugging within VFrame; however, there are certain methods for finding errors within VFrame that were not located with Workbench, Designer, or Report Editor debugging tools.

Part 3: More model, layout, report basics

These sections introduce more "basics" for models, layouts, and reports.

More layout basics

Workareas, pages, borders, lines, multimedia.

- **'25. Workareas (tabs) (D)' (page 71)**.
- **'26. Pages (in workarea frame) (D)' (page 72)**.
- **'27. Border, Line, Multimedia (D)' (page 73)**.

More report basics

Headers and footers, character and paragraph formats, multiple pages, page layouts.

- **'28. Report formatting (RE)' (page 76)**.

Data entry

Techniques for controlling data entry and display of results.

- **'29. Specifying default values for attributes (W)' (page 81)**. Demonstrates how to set default values for an attribute.
- **'30. Check value assigned to attribute (W)' (page 82)**. Demonstrates how to check the value of an attribute.
- **'31. Data indicator (W,D)' (page 84)**. Indicating when data input is required.
- **'32. Use Combo box / radio button group to select attribute values (W,D,RE)' (page 86)**. Demonstrates how an attribute's value can be selected from a range of valid values from a table.
- **'33. Select attribute value from multiple 2-column tables (W,D,RE)' (page 89)**. Demonstrates how attribute values can be selected from one of many simple 2-column tables.
- **'34. Select attribute value from single 3+-column table (W)' (page 92)**. Demonstrates how attribute values can be selected from a single table with multiple (3+) columns.
- **'35. Import a table from Excel database (Excel,W)' (page 93)**. Demonstrates how the value for an attribute can be input from an Excel table.
- **'36. Table contents direct from dBase file (Excel,W)' (page 95)**.

Dynamic data entry

- **'37. Dynamic calculation of default (W,D)' (page 96)**. Resetting Attribute1 to a default value if the

Attribute1 is specified as being dynamically dependent on Attribute2 and Attribute2 is changed.

- **‘38. Dynamic table contents (filter) (W,D)’ (page 98)**. Dynamically filtering the available values in a combo box or radio group based on the value of an attribute.
- **‘39. Dynamic label (W,D)’ (page 99)**. Labels that change dynamically.
- **‘40. Dynamic availability (W,D)’ (page 100)**. Dynamic availability. For example, an entry field can be dynamically disabled (displayed, but in grey and with data input disabled).
- **‘41. Dynamic visibility (W,D)’ (page 101)**. Demonstrates how an element’s visibility can be controlled by the value of an attribute. For example, an element can disappear if a value has been entered that makes that element irrevelant.

Dynamic data in the report

- **‘42. Dynamic layout elements in the report (RE)’ (page 103)**. For example, if an element is not visible in the layout, then it should probably not be included in the report either.

Dynamic arrays of data

Demonstrated with an example that generates interest and capital entries for each year.

- **‘43. Indexes and recursion (W,D)’ (page 105)**.
- **‘44. Grid (W,D)’ (page 107)**.
- **‘45. Graphics element (D)’ (page 110)**.

Dynamic arrays of data in the report

Special considerations are necessary for including arrays of data of varying size in the report.

- **‘46. Dynamic arrays in the report ??’ (page 112)**.

Part 4: Multiple products (product trees), user-defined functions and inclusion

These chapters introduce the concepts of **multiple products**, **product trees** and **inclusion**. **User-defined functions** are also introduced.

A typical insurance product may contain several subproducts (such as health, home, auto, etc.). These subproducts are modelled as product trees in the Workbench.

The conditions in which a product in a tree is included in the calculation of the premium is specified by the product inclusion type. The following inclusion types exist:

- **Mandatory**
- **Mutually exclusive**
- **Optional**
- **Multiply inclusive**

These chapters also demonstrate how to include product trees and inclusion in the **layout** and **report** (and introduce new layout and report functionality).

Product tree with mandatory products

Mandatory products are required. For example, for an auto insurance product, the subproduct auto liability insurance would be mandatory.

- **‘47. Create multiple subproducts (W)’ (page 113)**. Demonstrates how to create multiple subproducts for the overall product.
- **‘48. Create multiple subproducts in layout (D)’ (page 114)**. Demonstrates 1 way of displaying multiple products in a layout.
- **‘49. Add multiple subproducts to report (RE)’ (page 117)**. Demonstrates how to display multiple products in a report.

User-defined functions

Functions are similar to properties, with the exception that a function can appear in several trees simultaneously, whereas a property can only appear in the branch where it is defined and higher up in the tree (not in parallel branches).

- **‘50. Create functions (W)’ (page 124)**. Demonstrates how functions can be used to generate attribute values (using the today, years, and lookup functions).

Mutually exclusive products

Mutually exclusive products cannot be sold together. For example, subproducts for group insurance and individual insurance would be mutually exclusive.

- **‘51. Add mutually exclusive products in model (W)’ (page 127)**. Demonstrates how to specify mutually exclusive subproducts.
- **‘52. Add mutually exclusive products in layout (W, D)’ (page 129)**. Demonstrates 1 way of display-

ing mutually exclusive products in a layout.

- **'53. Add mutually exclusive products in Report (RE)' (page 132)**. Demonstrates how to display mutually exclusive products in a report.

Optional products

Optional products can be sold if desired. For example, hail damage insurance for an auto policy would probably be optional.

- **'54. Add optionally inclusive products to model (W)' (page 133)**. Demonstrates how to specify optionally inclusive subproducts.
- **'55. Add optionally inclusive products in layout (D)' (page 135)**. Demonstrates how to display optionally inclusive subproducts in a layout.
- **'56. Add optionally inclusive products to Report (RE)' (page 137)**. Demonstrates how to display optionally inclusive products in a report.

Multiply inclusive products

Multiply inclusive products can be included in a single policy more than once. For example, an auto dealer might insure every car on his lot. The actual premium for each car would be calculated individually (based on information enterd about the car) and added to the total premium.

- **'57. Add multiply inclusive products to model (W)' (page 138)**. Demonstrates how to specify multiply inclusive subproducts.
- **'58. Add multiply inclusive products in layout (D)' (page 141)**. Demonstrates how to display multiply inclusive subproducts in a layout.
- **'59. Add multiply inclusive products in layout as a Grid list (W, D)' (page 147)**. Demonstrates how to display multiply inclusive subproducts in a layout as a grid list.
- **'60. Add multiply inclusive products in layout as an Extended Grid list (D)' (page 150)**. Demonstrates how to display and **enter** multiply inclusive subproducts in a layout with an **extended** grid list.
- **'61. Add multiply inclusive products in layout as a Graphics element (D)' (page 154)**. Demonstrates how to display multiply inclusive subproducts in a graphics element.
- **'62. Add multiply inclusive products in Report (RE)' (page 155)**. Demonstrates how to print information about multiply inclusive products in a report.

Part 5: Objects, Events, Compensations, Components

These chapters introduce the concepts of **Objects**, **Events**, **Compensations** and **Components**, which can be used to make the product model much more descriptive, self-explanatory and easy to maintain.

These chapters also demonstrate how to include objects, events, compensations, and components in the **layout** and **report**.

Objects

Objects are the actual objects that are insured (such as Person, House, Car, etc.).

- **'63. Add Objects to the product tree (W)' (page 158)**. All information about the Person object in the previous examples is now moved to an Object.
- **'64. Add Objects to the layout (D)' (page 161)**. Demonstrates the minor modifications required for the layout.

Events

Events are those events that an object can experience and is insured against (such as personal Injury, home fire, auto hail damage, etc.).

- **'65. Add Events to the product tree (W)' (page 162)**. Demonstrates how to add multiple events with multiple inclusion.
- **'66. Add Events to the layout (D)' (page 164)**. Demonstrates how to add multiply inclusive events (which are subnodes of multiply inclusive objects). A more complex example, but introduces some interesting aspects of using the Designer and Workbench together.
- **'67. Add Events to Report (RE)' (page 167)**. Demonstrates how to includes the multiply inclusive events in a report. Some special considerations must be observed for nested multiple inclusions.

Compensations

Compensations are the types of (monetary) compensations that an object receives for an event (such as lost wage compensation in the event of personal injury, hotel expenses in the event of a home fire, repair costs in the event of hail damage, etc.).

- **'68. Add Compensations to the product tree (W)' (page 170)**. Demonstrates how to add multiple compensations with multiple inclusion.

- **‘69. Add Compensations to the layout (D)’ (page 173)**. Demonstrates how to add multiply inclusive compensations
- **‘70. Add Compensations to Report (RE)’ (page 177)**. Demonstrates how to add multiply inclusive compensations in a report.

Components

Sometime an object, event or compensation can appear several times in a product tree. In such cases, the object, event, or compensation is specified as a **component**. For example, compensation "Hotel expenses" could be included in the product tree for several different events: Home fire, earthquake, flood, etc. Specifying the compensation as a component allows changes made to one instance of the component to be effective wherever the component appears.

- **‘71. Components ??’ (page 180)**. Demonstrates how to create components.

Conclusion

- **‘72. Conclusion’ (page 186)**. Reviews what you have learned and where to go from here.

2. Using the included Zyx example files

This chapter describes how to use the .pms, .vpl, .cat, .ini, and .xls files included with this tutorial (on the CD ROM in directory \Doku\vpms) to start the tutorial at any chapter.

2.1. Example

If you want to start at ‘55. Add optionally inclusive products in layout (D)’ (page 135), then do the following:

2.1.1. Workbench model (.pms) file

2.1. If file **D:\Doku\vpms\Zyx49.pms** (the completed model at the end of chapter 49) exists: Copy to **C:\VPL_Apps\Zyx\Zyx.pms** (overwrite file if it exists).

2.1.2. Designer .vpl file

2.2. If file **D:\Doku\vpms\Zyx49.vpl** (the completed layout at the end of chapter 49) exists: Copy to **C:\VPL_Apps\Zyx\Zyx.vpl** (overwrite file if it exists).

2.1.3. IFOS RE .cat file

2.3. If file **D:\Doku\vpms\Zyx49.cat** (the completed model at the end of chapter 49) exists: Copy to **C:\VPL_Apps\Zyx\print\Zyx.cat** (overwrite file if it exists).

2.1.4. VFrame consult.ini file

2.4. Replace any content for the Zyx application in **C:\VPMS\VFrame\menu\consult.ini** with the contents of **D:\Doku\vpms\ZyxConsult##.ini** (where "##" is the greatest number less than 50). NOTE: Do not replace the entire contents of the .ini file (the file may contain configuration information for other consultations).

2.1.5. VFrame xplconf.ini file

2.5. Replace any content for the Zyx application in **C:\VPMS\VFrame\menu\xplconf.ini** with the contents of **D:\Doku\vpms\ZyxXplconf##.ini** (where "##" is the greatest number less than 50). NOTE: Do not replace the entire contents of the .ini file (the file may contain configuration information for other consultations).

You can now start the tutorial at ‘55. Add optionally inclusive products in layout (D)’ (page 135).

3. Specify extension defaults for VPMS files (Windows NT)

Certain file extensions are associated with files created by VPMS tools. This chapter describes how to specify the associations will enable the file to be opened in the required VPMS tool by simply double-clicking on the file.

In this chapter you will specify extensions for:

- Workbench models (.pms)
- Runtime models (.vpm)
- Designer layouts (.vpl)
- Runtime layouts (.vpl)
- IFOS reports (.cat).

3.1. Model (.pms) files

3.1. Display the CD ROM directory **\Doku\Vpms**.

3.2. Press and hold the **Shift** key.

3.3. Right-click on any **.pms** file. A pop-up menu appears.

3.4. Select **Open with...**. The **Open with** dialog appears.

Figure 3.1. Open with dialog (Windows NT)

3.5. Click **Other**. The **Open with...** dialog appears.

3.6. Double-click on **C:\VPMS\Wbench\Vpms32.exe**. The entry **Vpms32** is added to the program list.

3.7. Check the checkbox **Always open files of this type with this program**.

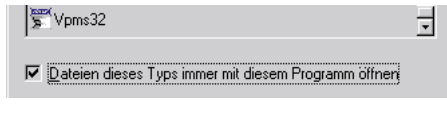


Figure 3.2. Checkbox for always opening file type with program

3.8. Click **OK**. The .pms file is opened in the Workbench.

3.9. Close the Workbench.

3.2. Runtime (compiled) model (.vpm) files

3.10. Specify that **.vpm** files are **always opened with** program **C:\VPMS\Wbench\Vpmste32.exe**.

3.11. Close the **Workbench test** dialog.

3.3. Layout (.vpl) files

3.12. Specify that **.vpl** files are **always opened with** program **C:\VPMS\Designer\vds.exe**.

3.13. Close the **Designer** dialog.

3.4. Runtime (compiled) layout (.vpc) files

3.14. Specify that **.vpc** files are **always opened with** program **C:\VPMS\Desiger\vds_tx32.exe**.

3.15. Close the **Designer test** dialog.

3.5. Report (.cat) files

3.16. Specify that **.cat** files are **always opened with** program **C:\VPMS\Wbench\ifosre.exe**.

3.17. Close the **IFOS Report Editor** dialog.

4. VFrame menus (Notepad)

This chapter describes how the VFrame menus are implemented. These menus can be changed by simply editting the .ini files.

For detailed information about the **VP/MS VFrame**, consult the **VFrame User's Guide**.

4.1. Open VFrame

4.1. From the **Start menu**: Select **Programs / VFrame32 / VFrame**. The VFrame dialog appears.

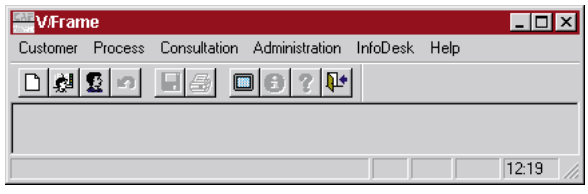


Figure 4.1. VP/MS VFrame main dialog

4.2. Menubar / toolbar

menubar.ini determines the content of the menubar and the toolbar.

4.2. Open **C:\vpms\vframe\menu\menubar.ini** (with notepad).

4.2.1. Menubar entries

Menubar entries are specified with the following format:

```
[Comments]
EINTRAGSART=1
EINTRAG=MenuItemText
EBENE=0
WIN_PROG=MENUE
PFAD_PROG_NAME=SubmenuIniFile.INI
```

For example, the menu item Kunde is specified with the following text:

```
[P1]
EINTRAGSART=1
EINTRAG=Kunde
EBENE=0
WIN_PROG=MENUE
PFAD_PROG_NAME=MENUKUND.INI
```

Figure 4.2. VFrame menubar

4.2.2. Toolbar entries

Toolbar entries are specified with the following format:

```
[Comments]
EINTRAGSART=0
EINTRAG=x
TOOLBAR=-1;IconNumber;IconToolTipText;1
WIN_PROG=INTERN
PFAD_PROG_NAME=vfkdv32d.dll_Routine
STATUSTEXT=VFrameStatusBarText
```

For example, the toolbar entry Kunde neu is specified with the following text:

```
[TOOLBAR]
EINTRAGSART=0
EINTRAG=x
TOOLBAR=-1;10;Kunde neu;1
WIN_PROG=INTERN
PFAD_PROG_NAME=KUNDE_NEU
```

```
STATUSTEXT=Neuaufnahme eines Kunden
```

Figure 4.3. VFrame toolbar icon Kunde neu

4.3. Menubar submenus

The content of each menubar submenu is managed by a separate .ini file.

4.3. Open **C:\vpms\vframe\menu\menukund.ini** (with notepad) This is the .ini file for menubar entry **Kunde**.

4.3.1. Submenu entries

Submenu entries are specified with the following format:

```
[Comments]
EINTRAGSART=0
EINTRAG=SubMenuItemText [TabCharacter]AcceleratorKeyText
EBENE=0
WIN_PROG=INTERN
PFAD_PROG_NAME=vfkdv32d.dll_Routine
STATUSTEXT=VFrameStatusBarText
ACCEL=AcceleratorKeyCombination
```

For example, the **Kunde** submenu item **Neuaufnahme... Ctrl+N** is specified with the following text:

```
[KUNDE_NEUAUFNAHME]
EINTRAGSART=0
EINTRAG=Neuaufnahme... Ctrl+N
EBENE=0
WIN_PROG=INTERN
PFAD_PROG_NAME=KUNDE_NEU
STATUSTEXT=Neuaufnahme eines Kunden
ACCEL='N'+VK_CONTROL
```

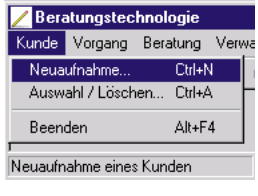


Figure 4.4. VFrame menu item Kunde submenu item NeuAufnahme

5. Manage agents (VF)

VFrame provides built-in functionality for creating and managing agents.

In this chapter you will:

- Create an agent.
- View the agent information in the database of agents.

5.1. Create agent

5.1. From the VFrame main menu: Select **Verwaltung / Vermittlerangaben**. The **Auswahl Anwender** (select agent) dialog appears.

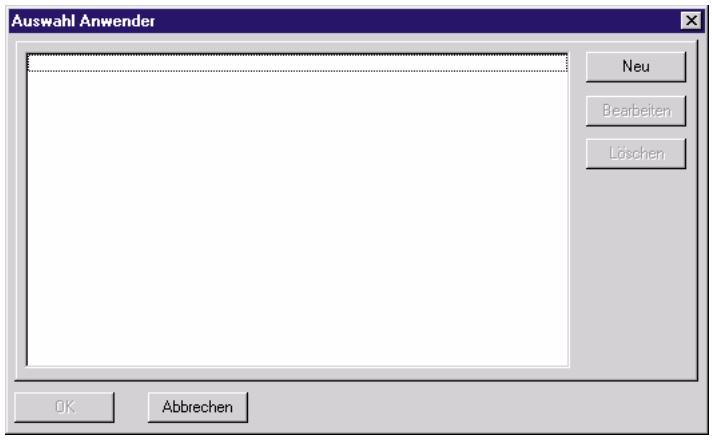


Figure 5.1. VFrame Auswahl Anwender (select agent) dialog

5.2. Click **Neu**. The following dialog appears for entering agent information:

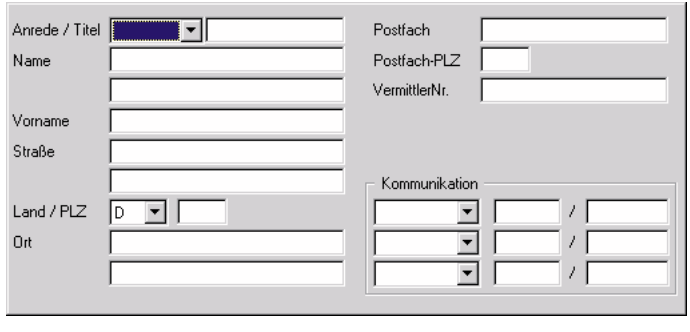


Figure 5.2. VFrame dialog for entering agent information

5.3. Enter information for the agent as shown below.

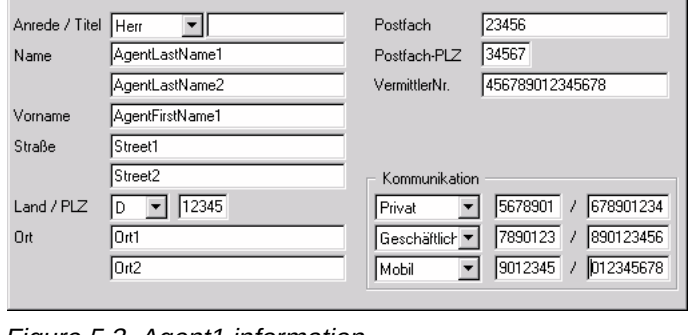


Figure 5.3. Agent1 information

5.4. Click **OK**. Agent1 appears in the list of agents.

Figure 5.4. Agent1 in the list of agents

5.5. Click **OK** to close the Auswahl Anwender dialog.

5.2. View agent info in anwender.dbf

All information for each agent is stored in the database anwender.dbf.

5.6. Open **C:\vpl_apps\data_bas\anwender.dbf**. Note the information for Agent1:

	A	B	C	D	E	F	G	H	
1	ANW_ID	ANREDE_KZ	ANREDE	TITEL	NAME1	NAME2	VORNAME	STRASSE1	STRASSE2
2	00000000000000000000	1	Herr		AgentLastName1	AgentLastName2	AgentFirstName1	Street1	Street2
3									

Figure 5.5. Agent1 info in anwender.dbf

6. Manage customers (VF)

VFrame provides built-in functionality for creating and managing customers.

In this chapter you will:

- Create a customer.
- View the customer information in the database of customers.
- Show a sublist of customers based on the last name.

6.1. Create customer

6.1. From the VFrame main menu: Select **Kunde / Auswahl/Löschen... Ctrl+A**. The **Auswahl Kunde / Vorgang** (select customer / vorgang) dialog appears.

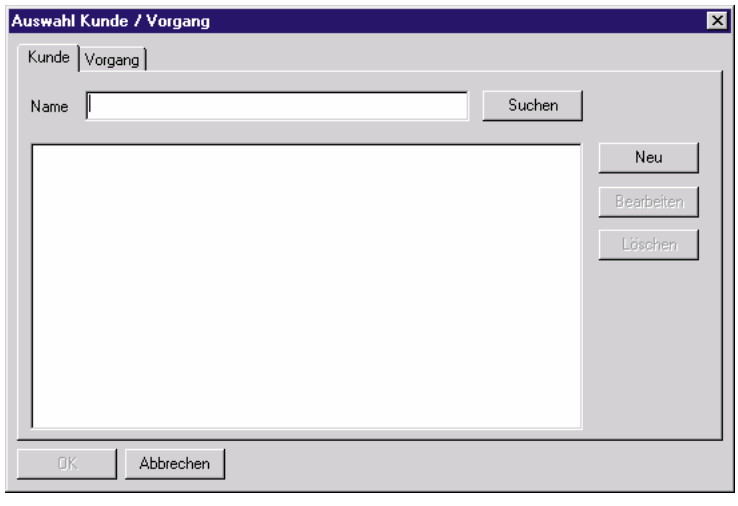


Figure 6.1. VFrame Auswahl Kunde / Vorgang (select customer / vorgang) dialog

6.2. Click **Neu**. The following dialog appears for entering customer information:

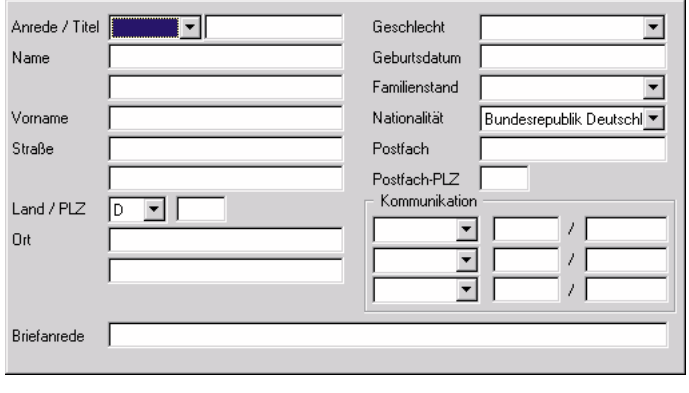


Figure 6.2. VFrame dialog for entering customer information

6.3. Enter information for the customer as shown below.

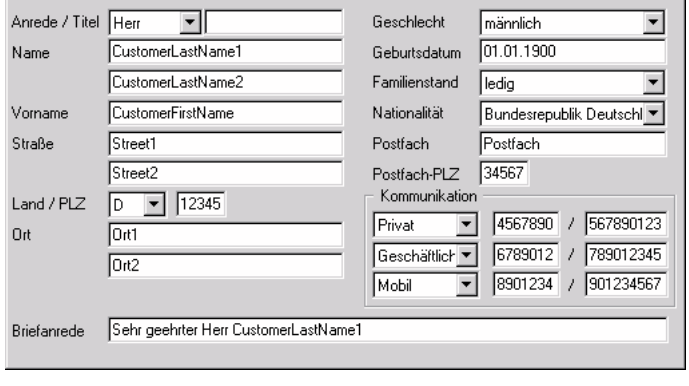


Figure 6.3. Customer1 information

6.4. Click **OK**. Customer1 appears in the list of customers.

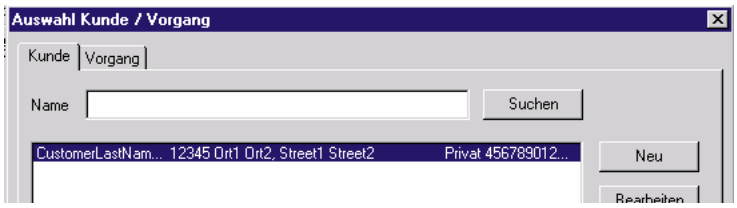


Figure 6.4. Customer1 in the list of customers

6.2. View customer info in person.dbf

6.5. Open **C:\vpl_apps\data_bas\person.dbf**. Note the information for Customer1:

	A	B	C	D	E	F	G
1	PER_ID	ANREDE_KZ	ANREDE	TITEL	NAME1	NAME2	VORNAME
2	00000000000000000000	1	Herr		CustomerLastName1	CustomerLastName2	CustomerFirstName
3							

Figure 6.5. Customer1 info in person.dbf

6.3. Search for customers

You can search the customer database for all customers whose last name (first part) contains a certain string.

Note: To show all customers: Search for customers whose names contain null ("").

6.6. In the field **Name**: Enter **CustomerLastName1**.

6.7. Click **Suchen**. All customers whose last name (first part) contains "CustomerLastName1" are displayed.

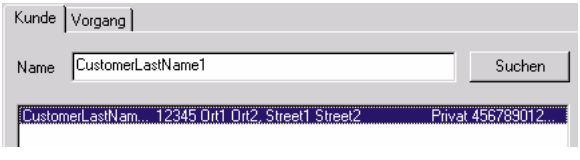


Figure 6.6. The list of customers whose last name (first part) contains "CustomerLastName1"

6.8. In the field **Name**: Enter **Cus**.

6.9. Click **Suchen**. All customers whose last name (first part) contains "Cus" are displayed.

6.10. In the field **Name**: Delete all (enter "").

6.11. Click **Suchen**. All customers are displayed.

6.12. Click **OK** to close the Auswahl Kunde / Vorgang dialog.

7. Create model (.pms) and runtime model (.vpm) (W)

The (design-time) model defines the business logic for the VPMS application. The run-time model provides this functionality within VPMS.

The design-time model is created in Workbench. The run-time model is generated from the design-time model with Workbench.

For detailed information about the **VP/MS Workbench**, consult the **Workbench User's Guide**.

In this chapter you will:

- Create a (design-time) model in Workbench (without saving)
- Test the contents of the Workbench
- Save the contents to design-time model file (.pms)
- Create a runtime model from the design-time model (the runtime model will be required by the Design layout)

7.1. Create design-time model (W)

7.1.1. Open Workbench

7.1. From the **Start menu**: Select **Programs / CAF VPMS / VPMS Workbench**. The Workbench dialog appears (the product information dialog will disappear after several seconds).

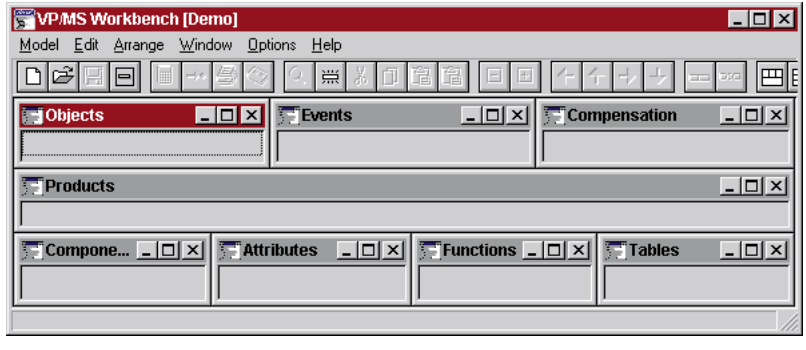


Figure 7.1. VP/MS Workbench main dialog

7.2. In the toolbar: Click on the **Show Products, Components, Attributes, Functions, Tables** icon ( ). The Workbench only displays those windows.

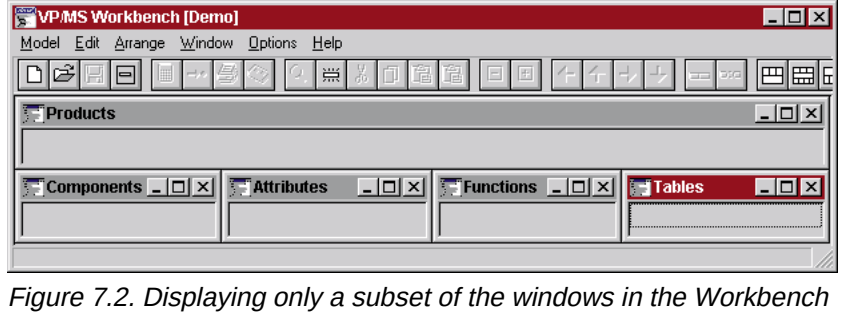


Figure 7.2. Displaying only a subset of the windows in the Workbench

7.1.2. Create product Product1 (W)

7.3. In the sub-window **Product**: Right-click. A popup window **New** appears:

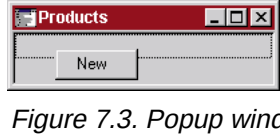


Figure 7.3. Popup window new in the Product window in the Workbench

7.4. Click on **New**. A new product (with no name) is created and selected in the Product window.

Figure 7.4. New product (with no name) in the Products window in the Workbench

7.5. Type **Product1**.

Figure 7.5. Product Product1 in the Workbench

7.1.3. Create attributes a_Persons, a_Weeks (W)

7.6. In the sub-window **Attributes**: Create an attribute named **a_Persons**.

7.7. In the sub-window **Attributes**: Create an attribute named **a_Weeks**.

Figure 7.6. Attributes a_Persons and a_Weeks in sub-window Attributes

7.1.4. Create property p_Premium for product Product1 (W)

7.8. Double-click on **Product1**. The **Inspector** for **Product1** appears:

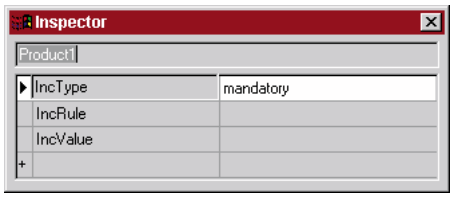


Figure 7.7. Inspector for Product1

7.9. Click with the mouse in the column to the right of the "+" sign. A text entry field is activated.

7.10. Enter **p_Premium**.

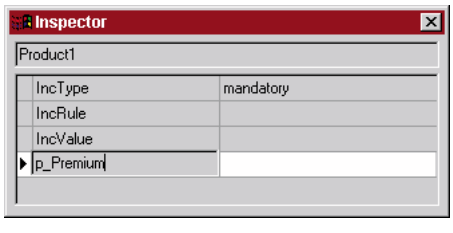


Figure 7.8. Parameter p_Premium for Product1

7.11. Click the **Enter** key. The entry field to the right of "p_Premium" obtains the focus.

7.12. Enter **a_Persons * a_Weeks**.

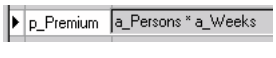


Figure 7.9. Parameter p_Premium definition

7.2. Test the (unsaved) design-time model (W,WT)

Clicking the Test Product Model icon in Workbench causes the current contents of the model in the Workbench to be tested with the VPMS Test tool (c:\vpms\wbench\vpmste32.dll).

7.13. Click on the **Test Product Model** icon ( ) in the Toolbar. The **Test** dialog appears.

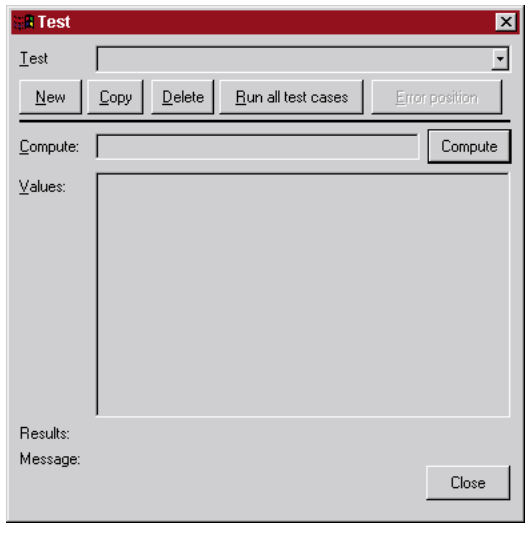


Figure 7.10. Test dialog

7.14. In the entry field **Compute**: Enter **p_Premium**.

7.15. Click on button **Compute**. Note that **a_Persons** appears in the **Values** box.

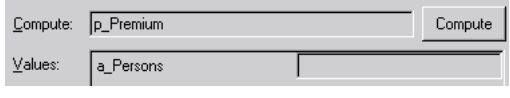


Figure 7.11. p_Premium and a_Persons in the Test dialog

7.16. Set **a_Persons** to **3**.

7.17. Click on button **Compute**. Note that **a_Weeks** appears in the **Values** box.

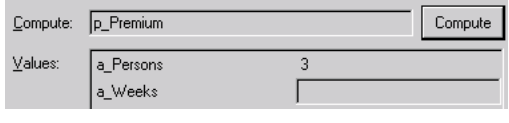


Figure 7.12. a_Weeks in the Test dialog

7.18. Set **a_Weeks** to **4**.

7.19. Click on button **Compute**. Note that field **Results:** displays **12**.

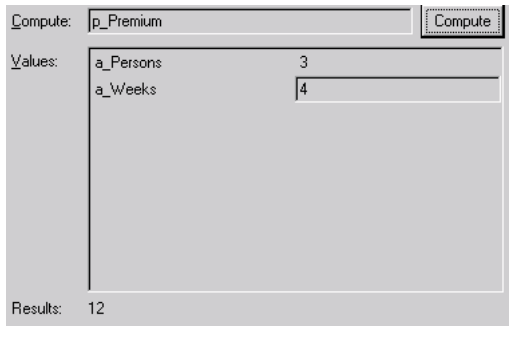


Figure 7.13. p_Premium result in the Test dialog

7.20. Click **Close**.

7.3. Save the design-time model (W)

7.21. Create directory **C:\VPL_Apps\Zyx**.

7.22. From the main menu: Select **Model / Save as...**.

7.23. Select directory **C:\VPL_Apps\Zyx**.

7.24. Specify the filename as **Zyx**.

7.25. Click **Save**. The **Save model** dialog appears with the version number, user, and comment for this version.

7.26. Click **OK**. The model is saved (Zyx04.pms).

7.4. Generate runtime model (W)

7.27. In the **Toolbar**: Click on the **Generate runtime module** icon ( ). The **Create runtime** dialog appears.

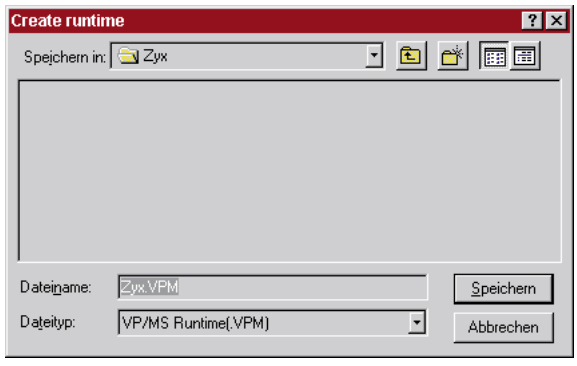


Figure 7.14. Workbench Create Runtime dialog

7.28. Click **Save** to generate the runtime model and save as **C:\VPL_Apps\Zyx\Zyx.vpm**.

8. Create layout (.vpl) and runtime layout (.vpc) (D)

The layout provides the user-interface to the business logic in the run-time model(s). The design-time layout is created in Designer. The run-time layout is generated from the design-time layout with Designer.

For detailed information about the **VP/MS Designer**, consult the **Designer User's Guide**.

In this chapter you will:

- Create a (design-time) layout
- Save the design-time layout / create a runtime layout from the design-time layout
- Test the runtime layout

8.1. Create layout (D)

8.1.1. Open Designer with new layout

8.1. From the **Start menu**: Select **Programs / CAF VPMS / VPMS Designer**. The Designer dialog appears.

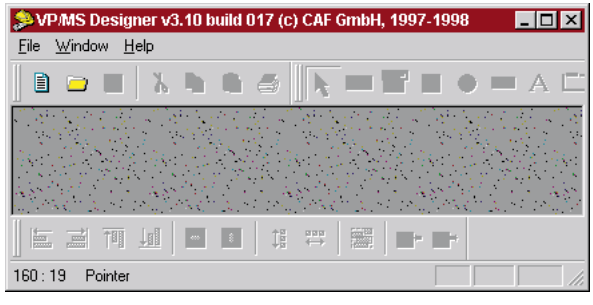


Figure 8.1. VP/MS Designer main dialog

8.2. From the main menu: Select **File / New**. A **VP/MS Designer window** and an **Inspector** appear.

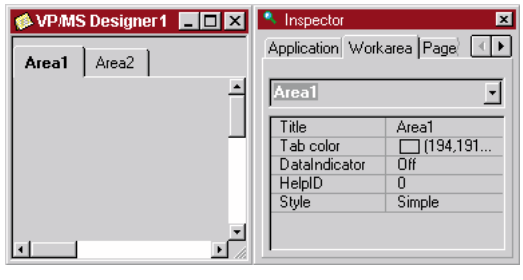


Figure 8.2. Designer window and Inspector in the Designer main dialog

Note: If the Inspector is not visible: From the main menu: Select **Options / Inspector**.

8.1.2. Specify product model (D)

Specifying the product model makes the attributes from the model available within the Designer.

8.3. In the **Inspector**: In tab **Application** (if not visible: scroll with the arrow buttons): Click on the **entry field** beside **Productmodel**. An open file button appears.

Figure 8.3. Directory button for selecting Productmodel

8.4. Click on the **Open File button**. The Open dialog appears.

8.5. Double-click on **C:\VPL_Apps\Zyx\Zyx.vpm**.

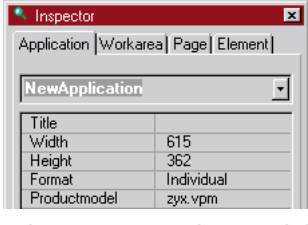


Figure 8.4. Productmodel selected in the Inspector

8.1.3. Add labels and attributes to layout (D)

8.6. In the **Tool Bar**: Click on the **New label** icon (A). Note: If the Tool Bar is not visible: From the main menu: Select **Window / View Tools**.

8.7. Click with the mouse in the **Designer** window. Note that a **label** appears.

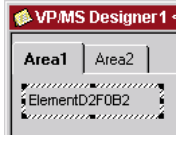


Figure 8.5. New label in the layout

8.8. Note in the **Inspector**: The tab **Element** property **Title** is selected. Change **Title** to **Insurance policy**.

8.9. Change the name of the **element** to **Label1**.

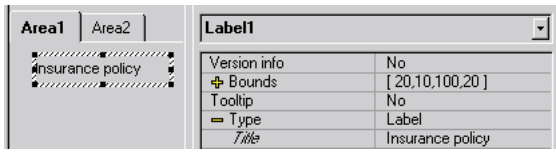


Figure 8.6. Defining Label name and property title in the Inspector

8.10. Add a **Label** with:

8.10.1. **Name** as **Label2**.

8.10.2. **Title** as **Number of persons**.

8.11. Add an **Edit field** ( ).

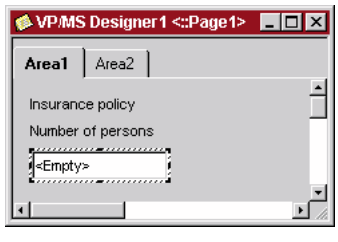


Figure 8.7. Edit field in layout

8.12. Set the **Name** of the edit field to **EF_Persons**.

8.13. In the **Inspector**: In tab **Element**: Click on **Source**. A **combo box** with available attributes from the model currently selected in the Workbench (Zyx.pms) appears:

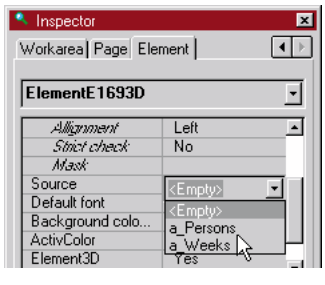


Figure 8.8. Combo box in Inspector with available attributes for the Edit Field

Note: If the Workbench attributes do not appear: Make sure that Zyx.pms is open in the Workbench and is the currently selected model (several models can be opened in the Workbench simultaneously). If the attributes still do not appear: From the Designer main menu: Select **Options / Workbench synchronisation**.

8.14. From the **combo box**: Select **a_Persons**.

8.15. Add a **Label** with:

8.15.1. **Name** as **Label3**.

8.15.2. **Title** as **Weeks of coverage**.

8.16. Add an **Edit field** with:

8.16.1. **Name** as **EF_Weeks**.

8.16.2. **Source** as **a_Weeks**.

8.17. Add a **Label** with:

8.17.1. **Name** as **Label4**.

8.17.2. **Title** as **Premium**.

8.18. Add a **VP/MS result field** ( ) with:

8.18.1. **Name** as **RF_premium**.

8.18.2. **Source** as **p_Premium**. Note: p_Premium will not be displayed in the combo box for Source; it must be typed in.

Figure 8.9. Layout with labels, 2 edit fields and VP/MS result field

8.2. Save the layout / create runtime layout (D)

8.19. Save the layout as **C:\VPL_Apps\Zyx\Zyx.vpl** (Zyx05.vpl). This will also create the Zyx.vpc (runtime) and Zyx.vpd files. Ignore any warning messages about page or element names.

8.3. Test runtime layout (D,DT)

Clicking the Test Application icon in Designer causes the following:

- If the design-time layout has unsaved changes: The design-time layout is saved.
- If the design-time layout has been changed since the last runtime layout was generated: A runtime layout is generated from the design-time layout.
- The runtime layout is tested with the Designer test tool (c:\vpms\wbench\vds_tx32.exe).

8.19.1. In the **Designer**: Click on the icon **Test application** ( ). A simplified **Designer Test** dialog is opened.

Figure 8.10. Designer layout in Designer test mini-dialog (vds_tx32.exe)

8.20. For **Number of persons**: Enter **7**.

8.21. For **Weeks of coverage**: Enter **8**.

8.21.1. Click to change the focus. Note the result of **56**.

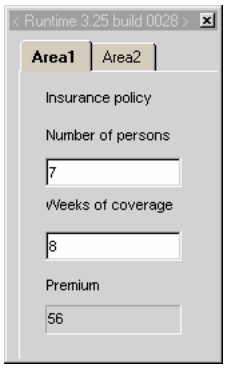


Figure 8.11. Designer test results

8.22. Close the test dialog.

9. Configure VFrame consultation Zyx (Notepad)

Layouts are used as the user interface for a customer consultation. A consultation must now be created that will use the layout (and model) created in the previous chapters.

In this chapter you will:

- Create an entry in the VFrame Beratung (consultation) menu for Zyx. This requires changes to consult.ini.
- Configure the XPL interface between VFrame and the runtime layout / model. This requires changes to xplconf.ini.

9.1. Configure the VFrame Beratung menu (consult.ini) (Notepad)

9.1. Open with Notepad the file **C:\Vpms\Vframe\menu\consult.ini**.

9.2. Add the following text to the end of the file:

```
[Zyx APPLICATION]
EINTRAGSART=0
EINTRAG=Zyx consultation
EBENE=0
PKEY=ZyxPKEY
WIN_PROG=10
PFAD_PROG_NAME=dummy
STATUSTEXT=Zyx status text
[this line ensures that above line ends with CR]
```

Description of the above lines:

- **[Zyx APPLICATION]**: A comment.
- **EINTRAGSART=0**: Selecting the menu entry sends a dll command (EINTRAGSART=2 is for a separator line).
- **EINTRAG=Zyx consultation**: The text that will be displayed in the VFrame submenu **Beratung**.
- **EBENE=0**: The level of the menu item (normally 0).
- **PKEY=ZyxPKEY**: A key that is used to match the entry in **consult.ini** with the entry in **xplconf.ini**.
- **WIN_PROG=10**: Specifies that selecting this menu item requires opening a compiled layout.
- **PFAD_PROG_NAME=dummy**:
- **STATUSTEXT=Zyx status text**: Status text shown in the status bar of VFrame.
- **[this line ensures that above line ends with CR]**: If a line does not end with a carriage return, then it is ignored in the consult.ini file. This comment line ensures that the previous line ends with a carriage return.

9.3. **Close and save the file** (Zyx08Consult.ini).

9.2. Configure the XPL interface (xplconf.ini) (Notepad)

9.4. Open with Notepad the file **C:\Vpms\Vframe\menu\xplconf.ini**.

9.5. Add the following text to the end of the file:

```
[Zyx APPLICATION]
TITEL=Zyx title
PKEY=ZyxPKEY
VPL_DATEI=Zyx.vpc
PFAD_VPL=%ROOT%\Zyx
PFAD_VPM=%ROOT%\Zyx
PFAD_DRUVOR=%ROOT%\Zyx\print
HILFEDATEI=rvint.hlp
PFAD_STRUC=%ROOT%\Zyx\struct
PFAD_HELP=%ROOT%\Zyx\help
PFAD_VORGANG_DATA=%ROOT%\Zyx\vorgang
DLL_Datei=%RT%
[this line ensures that above line ends with CR]
```

Description of the above lines:

- **[Zyx APPLICATION]**: Comment.
- **TITEL=Zyx title**:

- **PKEY=ZyxPKEY**: A key that is used to match the entry in **consult.ini** with the entry in **xplconf.ini**.
- **VPL_DATEI=Zyx.vpc**: Compiled layout file.
- **PFAD_VPL=%ROOT%\Zyx**: Directory of .vpc (compiled layout) file.
- **PFAD_VPM=%ROOT%\Zyx**: Directory of .vpm (compiled model) file.
- **PFAD_DRUVOR=%ROOT%\Zyx\print**: Directory of .cat (print) files.
- **HILFEDATEI=rvint.hlp**: Help file for consultation.
- **PFAD_STRUC=%ROOT%\Zyx\struct**: Directory of structure files.
- **PFAD_HELP=%ROOT%\Zyx\help**: Directory of help file for consultation.
- **PFAD_VORGANG_DATA=%ROOT%\Zyx\vorgang**: Directory of vorgang files.
- **DLL_Datei=%RT%**: DLL that interfaces to compiled layout (note the line RT=VDS_RT32.DLL).
- **[this line ensures that above line ends with CR]**: If a line does not end with a carriage return, then it is ignored in the xplconf.ini file. This comment line ensures that the previous line ends with a carriage return.

9.6. **Close and save the file** (Zyx08XplConf.ini).

10. Open consultation; Create vorgang (VF)

At this point you can finally open a consultation for a customer. The user interface for the consultation is the layout. The business logic is provided by the model.

This chapter also introduces vorgangs. A vorgang is a record of the data entered for a specific customer. The vorgangs for a customer are sequential: Vorgang N is created by:

- Opening vorgang N-1 for the customer
- Entering new data
- Saving the changes in a new vorgang (vorgang N)

The data displayed in the consultation reflects the net result of all actions recorded in the selected vorgang and all previous vorgangs. As an example, assume the following (where TYPE is customer data):

- Vorgang N is selected.
- In vorgang N: TYPE was not changed.
- In vorgang N-1: TYPE was set to -1.
- In vorgang N-2: TYPE was set to -2.

In vorgang N: The value displayed for TYPE is -1.

In this chapter you will:

- Open a consultation for Customer1
- Create vorgangs (enter data, save changes as vorgang) for Customer1
- Close the consultation
- Load a vorgang
- Analyze vorgang.dbf (database that contains vorgang info for all customers)
- Analzye vorgan_0.kon. This file specifies the interface between vfkdf32d.dll (VFrame dll) and vorgang.dbf.

10.1. Open consultation for Customer1 (VF)

10.1. Restart **VFrame**.

10.2. Select **Customer1** (select **Kunde / Auswahl/Löschen**; click **Suchen**; double-click on Customer1 in the customer list).

10.3. From the main menu: Select **Consultation / Zyx consultation**. The consultation (using the runtime model and layout) for Zyx appears:

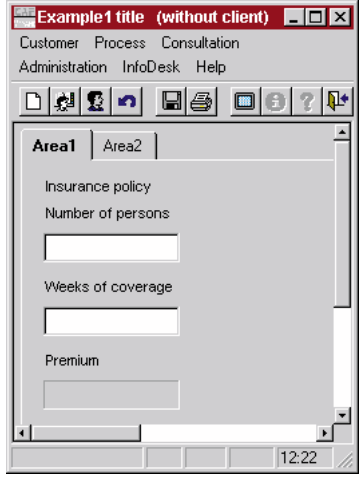


Figure 10.1. The runtime model and layout in VFrame consultation

10.2. Create vorgangs for Customer1 (VF)

10.2.1. Enter data

10.4. The entry field **Number of persons** has the focus. Enter **7**.

10.5. Press the **tab** key. The **Weeks of coverage** entry field has the focus.

10.6. Enter **8**.

10.7. Press the **tab** key. The **Number of persons** entry field has the focus and contains the value **56** (7

* 8) for **p_Premium**:

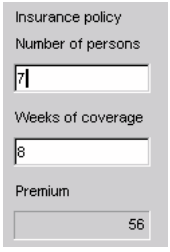


Figure 10.2. The calculated premium in VFrame

10.2.2. Save the entered data in a vorgang (VF)

10.8. Select **Vorgang / Tarifierung speichern**. The dialog **Vorgang speichern** appears:

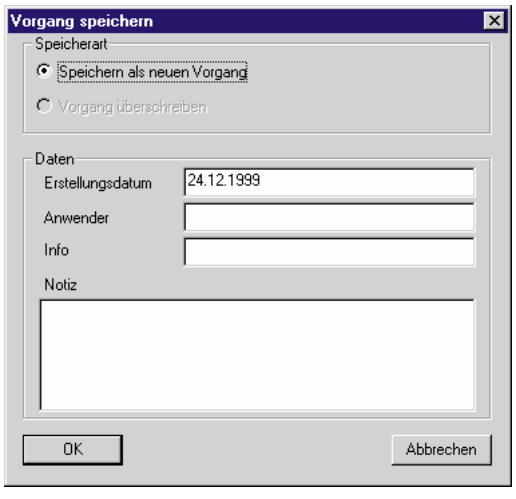


Figure 10.3. Dialog Vorgang speichern (vorgang save)

10.9. Enter the Vorgang information as shown:

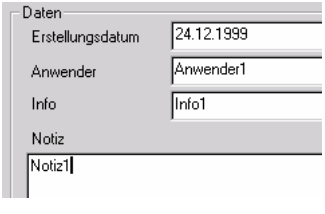


Figure 10.4. Vorgang information

10.10. Click **OK**.

10.2.3. Enter data; Save second vorgang

10.11. Enter for **Number of persons**: **8**.
10.12. Enter for **Weeks of coverage**: **9**.
10.13. Select **Vorgang / Tarifierung speichern**. The dialog **Vorgang speichern** appears.
10.14. Enter the Vorgang information: **Anwender1**, **Info2**, **Indiz2**.
10.15. Click **OK**.

10.3. Close consultation

10.16. Select **Vorgang / Tarifierung beenden**. The consultation is closed.

10.4. Load a vorgang

10.17. Select **Kunde / Auswahl/Löschen**. The **Auswahl Kunde/Vorgang** dialog appears.
10.18. Select **Customer1**.

10.19. Select tab **Vorgang**. Note the 2 vorgangs in the list.

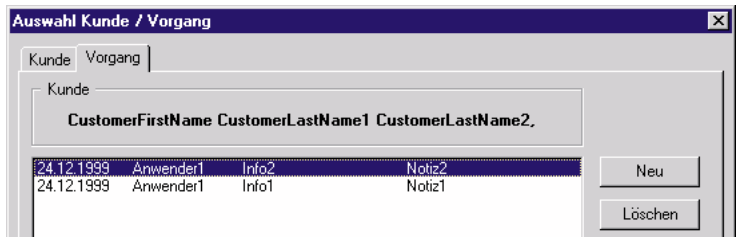


Figure 10.5. Tab Vorgang

10.20. Select the **first vorgang** (with Info1, Notiz1) (the vorgangs are listed in chronological order starting with the most recent vorgang at the top of the list).

10.21. Select **OK**. Note that the values (7, 8) in the consultation when the first vorgang was created reappear in the consultation.

10.5. Analyze vorgang info in vorgang.dbf

10.22. Open **C:\vpl_apps\data_bas\vorgang.dbf**. Note the information for the 2 vorgangs:

	A	B	C	D	E	F	G	H
1	VOR_ID	PER_ID	KMP_ID	ERST_DAT	INFO	NOTIZ	ANWENDER	BLOB_DATEI
2	19991224022016	00000000000000000000	*	24.12.99	Info1	Notiz1	Anwender1	BIG$@FYB.VOR
3	19991224022350	00000000000000000000	*	24.12.99	Info2	Notiz2	Anwender1	BIG#DOVL.VOR
4								
5								

Figure 10.6. Vorgang info in vorgang.dbf

Note the following:

- **VOR_ID** = YYYYMMDDHHMMSS when the vorgang was created.
- **PER_ID** = ID of the customer that the vorgang belongs to.
- **BLOB_DATEI** = name of the "blob" file (extension **.vor**; in directory **c:\vpl_apps\data_bas**) that contains information for the vorgang. The blob file contents are binary.

10.6. Analyze vfkdv32d.dll<->vorgang.dbf interface file vorgan_0.kon

vorgan_0.kon is created during VFrame installation and is the interface between the vfkdv32d.dll and vorgang database vorgang.dbf.

10.23. Open (in notepad) ASCII text file **c:\vpl_apps\data_bas\vorgan_0.kon**.

The list of variables ("VORGINR = VOR_ID", etc.) has the following format:

```
[vfkdv32d.dll variable] = [vorgang.dbf column name]
```

Note: If the dll variable and the vorgang.dbf column name are the same: vorgan_0.kon contains no entry for that variable / column name.

```
VORGINR          = VOR_ID
PRNINR           = PER_ID
XPL_DATEI        = BLOB_Datei
ERSTELLUNGSDATUM = Erst_Dat
;KMP_ID          = Kmp_ID
;INFO            = Info
;ANWENDER        = Anwender
;NOTIZ           = Notiz
```

11. Create report (.cat) (RE)

A report can be generated for the selected customer and vorgang. The information for the report is provided via the runtime layout and model.

A report can be printed to the screen or to a printer.

Note: Information in the customer database was entered without using the runtime layout or model. Therefore customer information is not available. A connection from the customer database to the runtime layout and model will be created later in this tutorial. This will enable customer information to be included in the printout.

For detailed information about the **VP/MS IFOS Report Viewer**, consult the **IFOS Report Viewer User's Guide**.

In this chapter you will:

- Open the IFOS report editor
- Enter the text for the report
- Mark the datafields in the text (the datafields are either report editor keywords (marked with #) or layout element names)
- Create a bereichskommando for the entire report (specifies the steuerdatei)
- Save the report (.cat)

11.1. Open IFOS RE (RE)

11.1. From the **Start menu**: Select **Programs / CAF Andwendungen / IFOS Report Editor**. The **IFOS Report Editor** dialog appears:

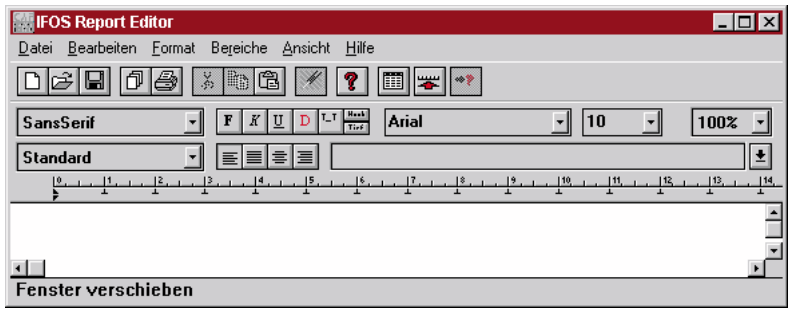


Figure 11.1. IFOS Report Editor main dialog

11.2. Enter text for report (RE)

11.2. Enter the following in the main text window:

```
This is page #Seite of report for Zyx.
Report created on #Datum at #Time.
The premium for EF_Persons persons for EF_weeks weeks is RF_Premium.
```

11.3. Select the entered text.

11.4. From the lower **combo box**: Select **Blocksatz** as the paragraph style for the entered text.

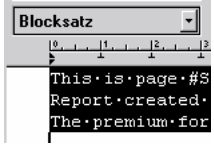


Figure 11.2. Blocksatz selected as the paragraph style

11.3. Mark text datafields (RE)

Datafields must be marked.

The current report contains the following types of datafields:

- RE reserved properties (with prefix "#")
- Layout element names (not attribute names)

11.5. Select text "**#Seite**".

11.6. Click on the **Datenfeldmarkierung an- oder ausschalten** icon (D) in the **Toolbar**.

11.7. Click elsewhere in the main text window. Note that the text **#Seite** is now red. This indicates that the text is a datafield.

11.8. Mark the following text as datafields:

11.8.1. **#Datum**

11.8.2. **#Time**

11.8.3. **EF_Persons**

11.8.4. **EF_Weeks**

11.8.5. **RF_Premium**

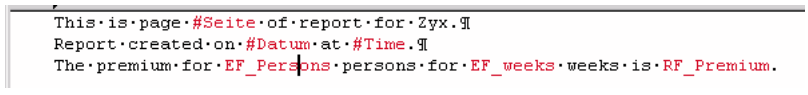


Figure 11.3. Text marked as datafields in the IFOS Report Editor main dialog

11.4. Enter bereichskommando for entire report (RE)

The bereichskommando for the entire report will specify the steuerdatei.

11.9. Click on the **Bereichskommando bearbeiten** button () in the **Werkzeuge** toolbar. The **Bereichskommando** dialog appears:

Figure 11.4. Bereichskommando dialog in the IFOS Report Editor

11.10. Enter the following text as the Bereichskommando:

```
Steuerdatei $DATEI_ROOT;
```

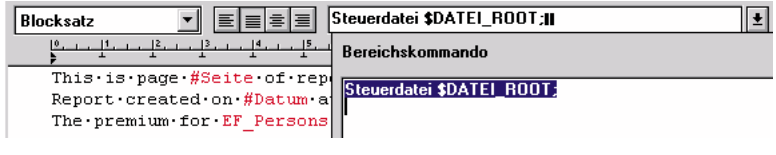


Figure 11.5. Bereichskommando text for the report

11.11. If the bereichskommando text was copied (using cut and paste) to the bereichskommando dialog: Retype the single quote (') marks in the bereichskommando text. **IMPORTANT:** Though appearing to visually be the same, the ASCII text actually copied to the Bereichskommando for single quote characters may not be the correct type of character.

11.5. Save as .cat file (RE)

11.12. Create directory **C:\VPL_Apps\Zyx\print**.

11.13. From the main menu: Select **File / Save as...**.

11.14. Save the file as **C:\VPL_Apps\Zyx\print\Zyx.cat** (Zyx06x.cat).

12. Create model attributes and tables for print support (W)

Certain attributes and tables are required in the model for print support.

In this chapter you will:

- Atribute a_druckauswahl
- Table t_druckauswahl
- Attribute a_druck1
- Table t_druck1

12.1. Create a_druckauswahl (W)

12.1. In the Workbench: Create attribute **a_druckauswahl**.

12.2. Doube-click on **a_druckauswahl** to open an **Inspector**.

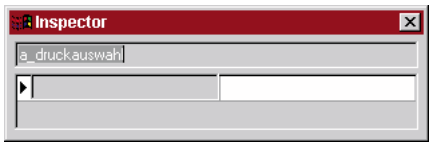


Figure 12.1. Inspector for a_druckauswahl

12.3. Enter property **table** with value **"t_druckauswahl"** (enter the quotation marks also).

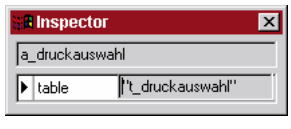


Figure 12.2. t_druckauswahl defined as table for a_druckauswahl

12.2. Create t_druckauswahl (W)

12.4. Create table **t_druckauswahl**.

12.5. Doube-click on **t_druckauswahl** to display contents in the **Inspector**.

12.6. Add key **DRUCK1** with value **Printouts1;Detail printouts1;Zyx.cat;0**.

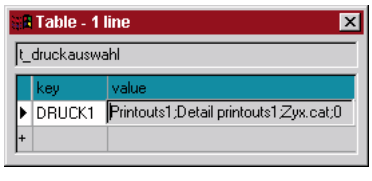


Figure 12.3. DRUCK1 defined in t_druckauswahl

12.3. Create a_druck1 (W)

12.7. Create attribute **a_druck1**.

12.8. Add property **table** with value **"t_druck1"** (enter the quotation marks also).

12.4. Create t_druck1 (W)

12.9. Create table **t_druck1**.

12.10. Doube-click on **t_druck1** to display contents in the **Inspector**.

12.11. Add key **1** with value **Zyx.cat**.

The following diagram show the attributes and tables:

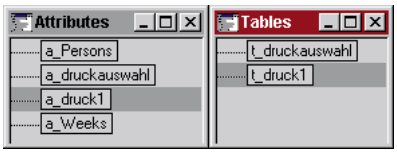


Figure 12.4. Defined attributes and tables for Zyx

12.12. **Save the model** (Zyx07.pms).

12.13. Generate a runtime module (**C:\VPL_Apps\Zyx\Zyx.vpm**) (overwrite the existing model).

13. Open consultation; Generate report (VF, RV)

A report can now be generated from VFrame. The report is viewed in the Report Viewer (c:\vpms\wbench\cafrg.exe).

In this chapter you will:

- Open the consultation for Customer1
- Print the vorgang info to the screen

13.1. Open consultation for Customer1

13.1. Restart **Vframe** (the runtime model was recompiled, required that VFrame is restarted).

13.2. From the main menu: Select **Kunde / Auswahl/Löschen**.

13.3. Select **Customer1**.

13.4. Select the last (top) **vorgang**.

13.5. Click **OK**. The consultation for customer1 is opened.

13.2. Print (VF,RV)

13.6. Select **Vorgang / Tarifierung drucken**. The **Druckauswahl** dialog appears with the printout and detailed printout options for Customer1:

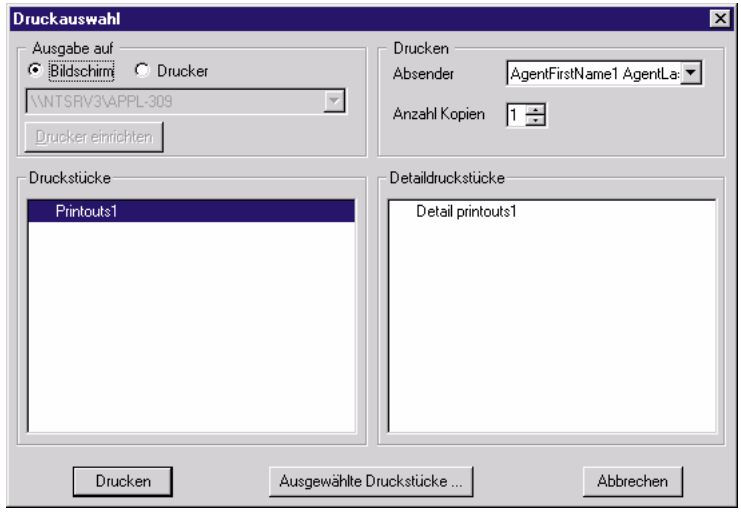


Figure 13.1. Druckauswahl options for Customer1

Note the **Absender** (agent) information in the upper-right of the dialog.

13.7. In the **Print to** box: Select **Screen**.

13.8. In the **Printouts** box: Select **Printouts1**.

13.9. In the **Detail printouts** box: Select **Detail printouts1**. Note the arrow beside **Printouts1**. This indicates that the printout is ready.

Figure 13.2. Printout ready indicator for Printouts1

13.10. Click **Print**. The **IFOS Report Viewer** main dialog appears with a view of the report.

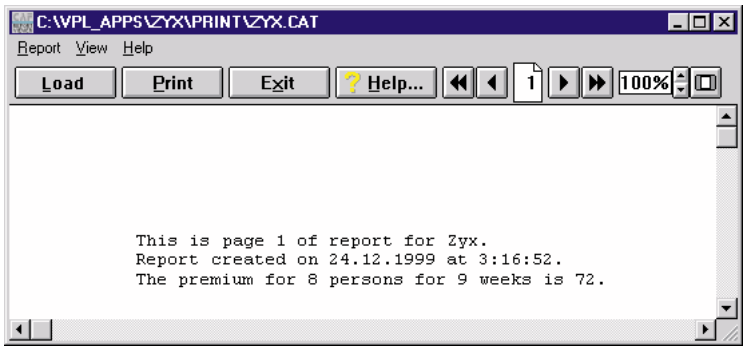


Figure 13.3. View of report in IFOS Report Viewer

13.11. Click on the **Drucken...** button to print the report to a physical (or logical) printer.

14. Add customer information to the consultation (W,D)

This chapter demonstrates how to add customer information to the consultation.

Pre-written model and layout components are used for the following reasons:

- Creating these components step-by-step at his point in the tutorial would be too complicated.
- The pre-written model and layout components conform to the standard format.

It is recommended, however, that you later analyze the contents of these files.

In this chapter you will:

- Include model Kunde.pms within model Zyx.pms
- Add Kunde entry fields to the layout
- Create DLL<->layout interface file Zyx.vpi
- Analyze vfkdv32d.dll<->person.dbf interface file kunde_0.kon (this file was created during VFrame installation)
- Open a consultation for Customer1 (note: the previous vorgangs are not compatible with the new model / layout, and therefore must be deleted)

14.1. Include model Kunde.pms within Zyx.pms (W)

Kunde.pms contains standard logic for controlling the appearance of the customer information in the model.

14.1. Copy **[CD ROM drive]:\doku\Zyx40_kunde.pms** to **C:\VPL_Apps\Zyx\kunde.pms**.

14.2. Open **Zyx.pms** in the Workbench.

14.3. Select **Model / Included**. The **Included models** dialog appears.

14.4. Click **Add**. The **Load model** dialog appears.

14.5. Double-click on **Kunde.pms**.

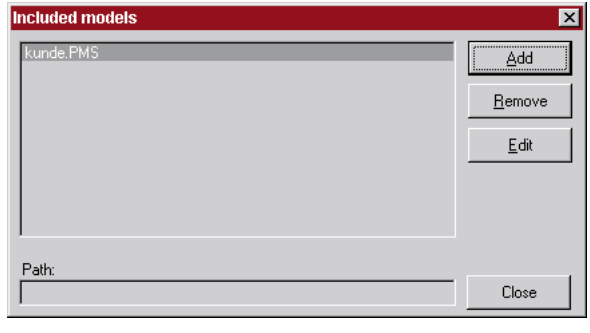


Figure 14.1. Kunde.pms in the Included Models dialog

14.6. Click **Close**.

14.7. Save the model (Zyx40.pms).

14.8. Create a runtime model. Note the tab for the included model **Kunde** after **Zyx** was compiled.

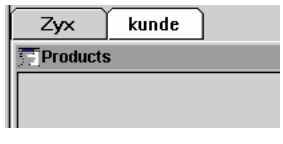


Figure 14.2. Kunde.pms tab in the Workbench

14.2. Add Kunde entry fields to the layout (D)

Zyx40_Kunde.vpl contains standard layout components for customer information in the layout.

14.2.1. Copy Zyx40_Kunde.vpl components

14.9. Open **Zyx40_Kunde.vpl**.

14.10. Move the mouse ot the upper-left corner of the layout.

14.11. Press and hold down the left mouse button.

14.12. Move the mouse to the lower-right of the screen.

14.13. Release the mouse button. Note that all layout components have been selected.

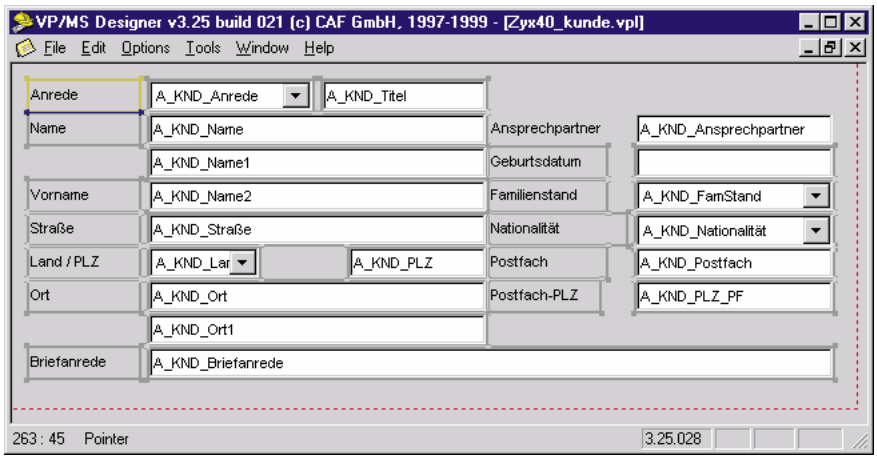


Figure 14.3. Selected layout components in Zyx40_Kunde.vpl

14.14. Press **Ctrl-C**.

14.2.2. Paste Zyx40_Kunde.vpl components to Zyx.vpl

14.15. Open **Zyx.vpl** (or if Zyx.vpl is already open in the Designer: select the Zyx.vpl window from sub-menu **Window**).

14.16. Select tab **Area2**.

14.17. Press **Ctrl-V**. Note that the layout components are copied to Zyx.vpl.

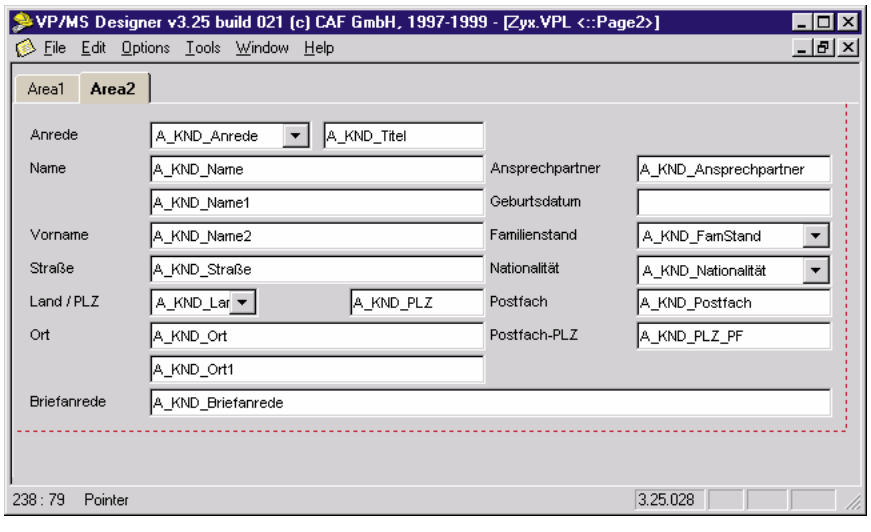


Figure 14.4. Contents of Zyx40_Kunde.vpl in Zyx.vpl

14.18. Save the layout (Zyx40.vpl).

14.3. Create DLL<->Layout interface file zyx.vpi

zyx.vpi is the interface between the vfkdv32d.dll and the runtime layout zyx.vpc.

14.19. Create ASCII text file **c:\vpl_apps\zyx\struct\zyx.vpi** with the following content:

```
; on left: dll variable
;   (if dllname <> dbname: dll variable shown on left in kunde_0.kon)
;   (if dllname = dbname: dll variable shown as column name in person.dbf)
; on right: designer element name (NOT model attribute name)
; make sure the last line ends in a cr !!!!

[PERSON]
PF=A_Postfach
PFPLZ=A_PLZPF
PNRPC=PER_ID
PTNAME=A_KND_Name
TITEL=A_KND_Titel
ANREDE_KZ=A_KND_Anrede
```

```
NZUSATZ=A_KND_Name1
VORNAME=A_KND_Vorname
LKZ=A_KND_Land
PLZ=A_KND_PLZ
ORT=A_KND_Ort
OT=A_KND_Ort1
Ort1=A_KND_Ort1nat
STRHSNR=A_KND_Straße
GEBDAT=A_KND_GebDat
```

14.4. Analyze vfkdv32d.dll<->person.dbf interface file kunde_0.kon

kunde_0.kon is created during VFrame installation and is the interface between the vfkdv32d.dll and customer database person.dbf.

14.20. Open (in notepad) ASCII text file **c:\vpl_apps\data_bas\kunde_0.kon**.

The list of variables ("PNRPC = PER_ID", etc.) has the following format:

`[vfkdv32d.dll variable] = [person.dbf column name]`

Note: If the dll variable and the person.dbf column name are the same: kunde_0.kon contains no entry for that variable / column name.

```
PNRPC     = PER_ID    ;Primary Key
PTNAME    = Name1     ;Nachname
NZUSATZ   = Name2     ;Name 2
ORT       = Ort1      ;Ort Teil 1
OT        = Ort2      ;Ort Teil 2
GEBDAT    = Geb_Dat   ;Geburtsdatum
STRHSNR   = Strasse1  ;Strasse und Hausnummer
PF        = Postfach  ;Postfach
PFPLZ     = Postf_PLZ ;PLZ Postfach
```

14.5. Open consultation in VFrame

14.5.1. Delete previous vorgangs

The previous vorgangs will not work with the new layout.

14.21. Restart **VFrame**.

14.22. Select **Kunde / Auswahl/Löschen**.

14.23. Select **Customer1**.

14.24. Select tab **Vorgang**.

14.25. Select the first **vorgang**.

14.26. Click **Löschen**.

14.27. Click **OK**.

14.28. Select the remaining **vorgang**.

14.29. Click **Löschen**.

14.30. Click **OK**.

14.5.2. Create new vorgang

14.31. Click **Neu**. The layout is displayed with the new vorgang.

14.32. Select **Area2**. Note the Customer1 information.

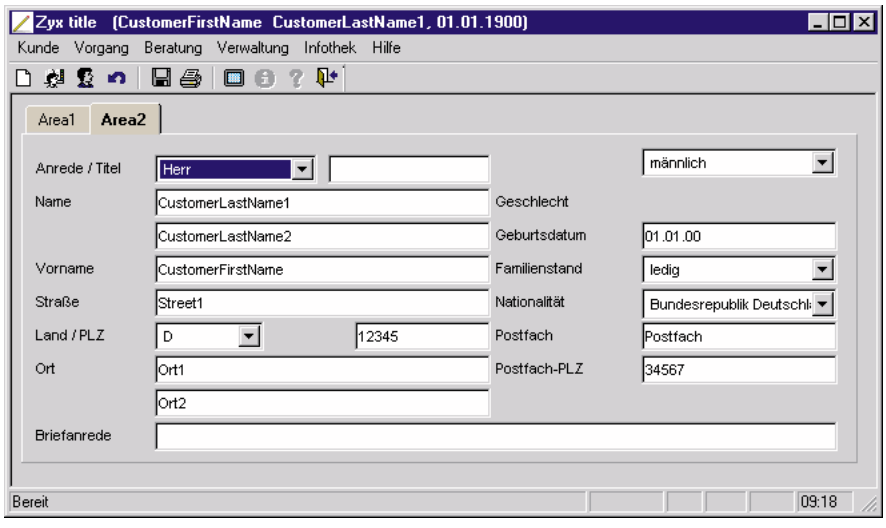


Figure 14.5. Customer information in VFrame layout

15. Add customer information to the report (RE,RV) ??

This chapter demonstrates how to add customer information to the report.
In this chapter you will:

- .

15.1. Add customer information to the report (RE) ??

Customer information is now available via layout elements and thus can now be added to the report.
15.1.

16. Add push buttons to the layout for accessing vfkv32d.dll dialogs

The customer information should not be editted in the VFrame layout (the changes would only be implemented in the vorgang, not in the person.dbf database). However, it would be convenient to access the customer information dialogs from the layout. This is possible by adding pushbuttons to the layout.

In this chapter you will:

- Create the required .ini file for each vfkv32d.dll function (dialog)
- Add a pushbutton for each vfkv32d.dll function. The pushbutton will call the required function with the help of the required .ini file.
- Add an entryfield and checkbox (required by the model logic)
- Test

16.1. Create .ini files for calls to vfkv32d.dll (Notepad)

16.1.1. CustomerNew ini file

16.1. Create **c:\vpl_apps\struct\zyx\kndneu.ini** (create a new customer) with the following content:

```
; This is configuration file for DLL-cal
[common]
; DLL and function names
DLL=VFkdV32D.dll
Method=KND_Neu
DLL16=VFkdV16D.dll
Method16=_KND_Neu
[input]
; Here visual element names described used as input parameters for DLL-call
0=$hwnd
1=Zuweisen
[output]
; Here visual element names are described used as results of DLL-call
0=PER_ID
1=A_KND_Anrede
2=Anrede_Text
3=A_KND_Titel
4=A_KND_Name
5=A_KND_Name1
6=A_KND_Vorname
7=A_KND_Straße
8=Strasse2
9=A_KND_Land
10=A_KND_PLZ
11=A_KND_Ort
12=A_KND_Ort1
13=A_KND_Geschlecht
14=Geschlecht_Text
15=A_KND_GebDat
16=A_KND_FamStand
17=FamStand_Text
18=A_KND_Nationalitaet
19=Staat_Text
20=A_Postfach
21=A_PLZPF
22=Briefanrede
23=KoArt_KZ1
24=KommArt1
25=Vorwahl1
26=Nummer1
27=KoArt_KZ2
28=KommArt2
29=Vorwahl2
```

```
30=Nummer2
31=KoArt_KZ3
32=KommArt3
33=Vorwahl3
34=Nummer3
35=iFax
36=iTelefon
```

16.1.2. CustomerSearch ini file

16.2. Create **c:\vpl_apps\struct\zyx\kndsuch.ini** (find a customer) with the following content:

```
; This is configuration file for DLL-cal
[common]
; DLL and function names
DLL=VFkdV32D.dll
Method=KND_Suchen
DLL16=VFkdV16D.dll
Method16=_KND_Suchen
[input]
; Here visual element names described used as input parameters for DLL-call
0=$hwnd
1=Zuweisen
[output]
; Here visual element names are described used as results of DLL-call
0=PER_ID
1=A_KND_Anrede
2=Anrede_Text
3=A_KND_Titel
4=A_KND_Name
5=A_KND_Name1
6=A_KND_Vorname
7=A_KND_Straße
8=Strasse2
9=A_KND_Land
10=A_KND_PLZ
11=A_KND_Ort
12=A_KND_Ort1
13=A_KND_Geschlecht
14=Geschlecht_Text
15=A_KND_GebDat
16=A_KND_FamStand
17=FamStand_Text
18=A_KND_Nationalitaet
19=Staat_Text
20=A_Postfach
21=A_PLZPF
22=A_Briefanrede
23=KoArt_KZ1
24=KommArt1
25=Vorwahl1
26=Nummer1
27=KoArt_KZ2
28=KommArt2
29=Vorwahl2
30=Nummer2
31=KoArt_KZ3
32=KommArt3
33=Vorwahl3
34=Nummer3
35=iFax
36=iTelefon
```

16.1.3. CustomerEdit ini file

16.3. Create **c:\vpl_apps\struct\zyx\kndbearb.ini** (edit a customer) with the following content:

```
; This is configuration file for DLL-cal
[common]
; DLL and function names
DLL=VFkdV32D.dll
Method=KND_Bearbeiten
DLL16=VFkdV16D.dll
Method16=_KND_Bearbeiten
[input]
; Here visual element names described used as input parameters for ; DLL-call
0=$hwnd
1=Zuweisen
2=PER_ID
[output]
; Here visual element names are described used as results of DLL-call
0=PER_ID
1=A_KND_Anrede
2=Anrede_Text
3=A_KND_Titel
4=A_KND_Name
5=A_KND_Name1
6=A_KND_Vorname
7=A_KND_Straße
8=Strasse2
9=A_KND_Land
10=A_KND_PLZ
11=A_KND_Ort
12=A_KND_Ort1
13=A_KND_Geschlecht
14=Geschlecht_Text
15=A_KND_GebDat
16=A_KND_FamStand
17=FamStand_Text
18=A_KND_Nationalitaet
19=Staat_Text
20=A_Postfach
21=A_PLZPF
22=A_Briefanrede
23=KoArt_KZ1
24=KommArt1
25=Vorwahl1
26=Nummer1
27=KoArt_KZ2
28=KommArt2
29=Vorwahl2
30=Nummer2
31=KoArt_KZ3
32=KommArt3
33=Vorwahl3
34=Nummer3
35=iFax
36=iTelefon
```

16.2. Add push buttons for dll call to layout (D)

16.2.1. CustomerNew pushbutton

16.4. In the Designer: Select workarea **Area2**.

16.5. In the **Tool Bar**: Click on the **New pushbutton** icon (). Note: If the Tool Bar is not visible: From the main menu: Select **Window / View Tools**.

16.6. Click with the mouse in the **Designer** window. Note that a **pushbutton** appears.

Figure 16.1. New pushbutton in the layout

16.7. In the **Inspector**: Change the name of the **pushbutton** to **P_ANT_Neu**.
16.8. Change property **Title** to **New**.
16.9. Change property **Action** to **DLL-Call**.
16.10. Click in the entry area for property **Dll-Description**. A pushbutton appears.
16.11. Click on the pushbutton. The **Open** dialog appears.
16.12. Double-click on **c:\vpl_apps\zyx\struct\kndneu.ini**.

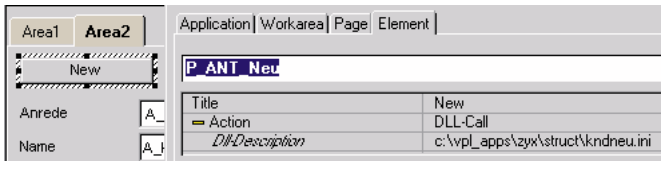


Figure 16.2. Pushbutton properties in the Inspector

16.2.2. CustomerSearch pushbutton

16.13. Add a **pushbutton** to the layout.
16.14. Change the name of the **pushbutton** to **P_ANT_Suchen**.
16.15. Change property **Title** to **Search**.
16.16. Change property **Action** to **DLL-Call**.
16.17. Click in the entry area for property **Dll-Description**. A pushbutton appears.
16.18. Click on the pushbutton. The **Open** dialog appears.
16.19. Double-click on **c:\vpl_apps\zyx\struct\kndsuch.ini**.

16.2.3. CustomerEdit pushbutton

16.20. Add a **pushbutton** to the layout.
16.21. Change the name of the **pushbutton** to **P_ANT_Bearbeiten**.
16.22. Change property **Title** to **Edit**.
16.23. Change property **Action** to **DLL-Call**.
16.24. Click in the entry area for property **Dll-Description**. A pushbutton appears.
16.25. Click on the pushbutton. The **Open** dialog appears.
16.26. Double-click on **c:\vpl_apps\zyx\struct\kndbearb.ini**.

Figure 16.3. 3 pushbuttons in the layout for dll calls

16.3. Add required entry field / checkbox to the layout (D)

The model contains logic for avoiding errors due to null or undefined values. The following elements are required in the layout to support this logic, since VFrame communicates directly with the runtime layout and not the runtime model.

Note: Normally (ie, not in a tutorial) these fields would be non-visible.

16.27. On workarea **Area2**: Add a **EntryField**.
16.28. Set **EntryField name** to **PER_ID**.
16.29. Set **EntryField** property **Source** to **PER_ID**.
16.30. Add a **CheckBox**.
16.31. Set **CheckBox name** to **Zuweisen**.
16.32. Set **CheckBox** property **Title** to **Zuweisen**.

16.33. Set **CheckBox** property **Source** to **Zuweisen**.

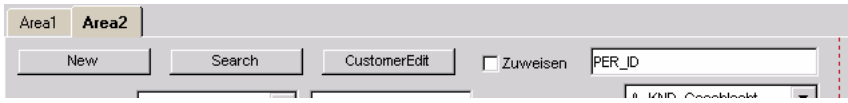


Figure 16.4. Layout with 3 Pushbuttons, EntryField and Checkbox

16.34. Save the layout.

16.4. Test (VF)

16.35. Open vorgang for Customer1.

16.36. Select workarea **Area2**.

16.37. Click on the **Edit** PushButton. The **Kunde** dialog appears.

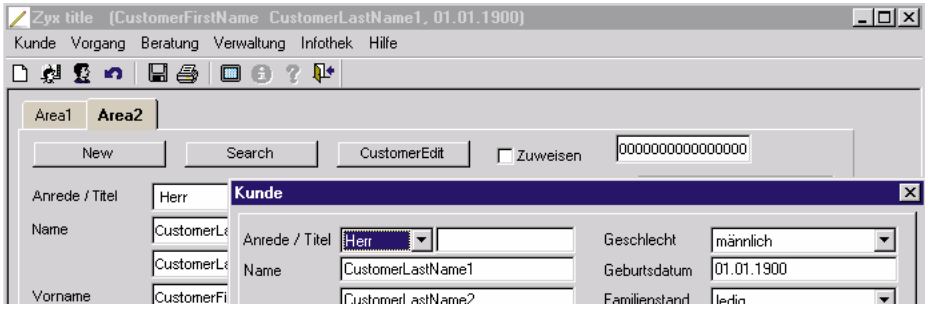


Figure 16.5. Auswahl Kunde dialog using Suchen pushbutton

16.38. Test the **New** and **Search** buttons.

17. Model versions (vpms.ini, .pms) (W)

17.1. Specifying default author and commentary for model versions (VPMS.ini) (W)

17.1. Add the following lines (or edit the lines if they already exist) in the file **C:\Winnt\VPMS.ini**:

```
Autor=ZYX
Kommentar=ZYX's vpms project
```

17.2. Save the file.

17.3. In the **Workbench**: Click **Save**. Note the default version information:

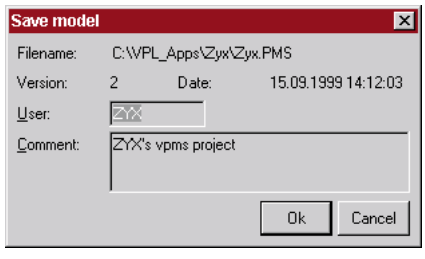


Figure 17.1. Default version information in the Workbench Save dialog

17.4. Click **OK**.

17.2. Load previous version of model (W)

17.5. From the menu: Select **Model / Version**. The **Versions** dialog appears:

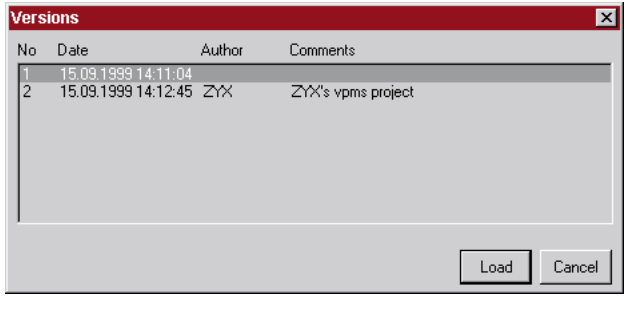


Figure 17.2. Versions dialog in the Workbench

17.6. Select the **previous version**.

17.7. Click **Load**. Note the version number in the title bar:

Figure 17.3. Version number in the Workbench title bar

17.3. Saving a previous version (overwriting newer versions) (W)

17.8. Click on the **Save** icon. A warning message **"Saving older version will overwrite newer version"** appears.

17.9. Click **OK** (the versions are the same). The **Save model** dialog appears.

17.10. Click **OK** (Zyx10.pms).

18. Testing the model (.pms, .vpm) (W, WT)

18.1. Basic test from within Workbench (W,WT)

18.1. Click on the **Test Product Model** icon ( ) in the Toolbar. The **Test** dialog appears.

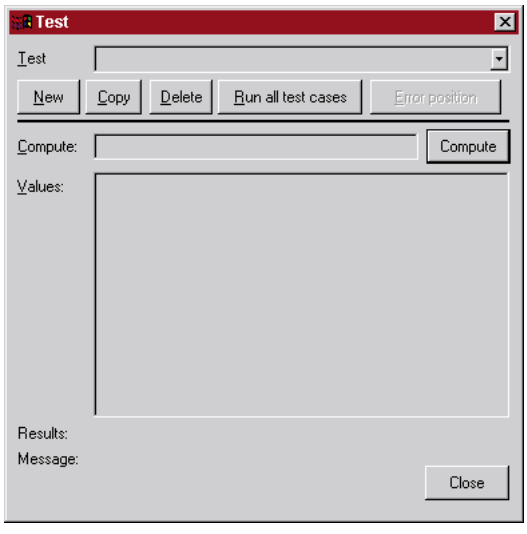


Figure 18.1. Test dialog

18.2. In the entry field **Compute**: Enter **p_Premium**.

18.3. Click on button **Compute**. Note that **a_Persons** appears in the **Values** box.

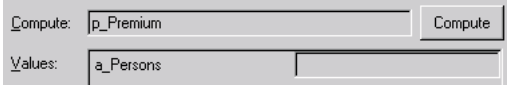


Figure 18.2. p_Premium and a_Persons in the Test dialog

18.4. Set **a_Persons** to **3**.

18.5. Click on button **Compute**. Note that **a_Weeks** appears in the **Values** box.

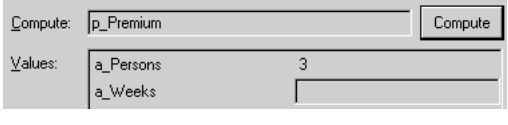


Figure 18.3. a_Weeks in the Test dialog

18.6. Set **a_Weeks** to **4**.

18.7. Click on button **Compute**. Note that field **Results:** displays **12**.

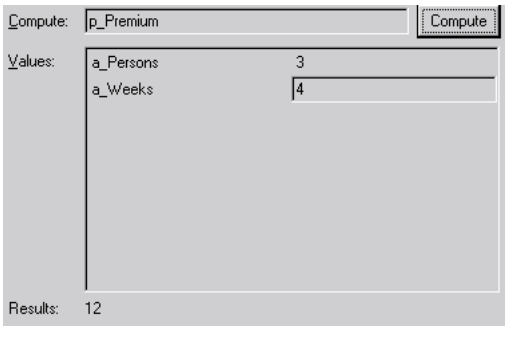


Figure 18.4. p_Premium result in the Test dialog

18.2. Modify model (W)

18.8. Change the definition for **Product1** property **p_Premium** to **a_Persons * a_Weeks + 1**.

18.3. Repeat basic test from within Workbench (W,WT)

18.9. Repeat the test above. The result is **13**.

18.4. Perform basic test outside Workbench (WT)

18.10. Double-click on **Zyx.vpm**. Repeat the test above. The result is **12**. The .pms and .vpm files have not been updated with the new definiton (with " + 1 ") for p_Premium. When testing within Workbench, any changes are incorporated in the test, even if the changes have not been saved to the model and/or runtime model.

18.5. Generate runtime; repeat basic test outside Workbench (W,WT)

18.11. Generate a runtime model (overwrite the previous model).

18.12. Double-click on **Zyx.vpm**. Repeat the test above. The result is now **13**.

18.6. Creating test case test1 (W,WT)

18.13. In the **Test** combo box: Enter **test1**.

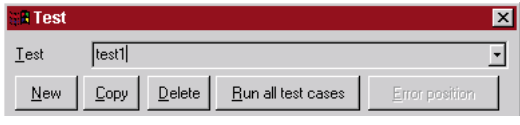


Figure 18.5. Specifying the name of a test in the Test dialog

18.14. In the **Test** dialog: Click on **New**. This save the previous test and clears the Test dialog for a new test. test1 can now be loaded from the test case combo box.

18.7. Creating test case test2 (W,WT)

18.15. In the entry field **Compute**: Enter **p_Premium**.

18.16. Click **Compute**.

18.17. For **a_Persons**: Enter **6**.

18.18. Click **Compute**.

18.19. For **a_Weeks**: Enter **7**.

18.20. Click **Compute**. The result of **42** is displayed.

18.21. In the **Test** combo box: Enter **test2**.

18.22. Click on **New** to save the test.

18.8. Run all test cases (W,WT)

18.23. In the **Test** dialog: Click on **Run all test cases**. The message box **"All test cases returned identical results"** appears.

18.24. Click **OK**.

18.25. Close the test dialog.

18.9. Open test outside Workbench (WT)

18.26. Double-click on **Zyx.vpm**. Note that the test cases are not available.

18.10. Unmodify model (W)

18.27. For **Product1**: Change the definition of **p_Premium** to **a_Persons * a_Weeks**.

18.11. Run all tests within Workbench (W,WT)

18.28. Click on the **Test product model** icon. The test dialog appears.

18.29. Click on **Run all test cases**. The following message appears: **"Test case 'test1' returned '13' and now '12'. Accept new value?"**.

18.30. Click **Yes**. The following message appears: **"Test case 'test2' returned '43' and now '42'. Accept new value?"**.

18.31. Click **No**.

18.32. Select **test1** from the **Test:** combo box. Note that the new value of **13** is displayed.

18.33. Select **test2** from the **Test:** combo box. Note that the old value of **42** is displayed.

18.34. Close the test dialog.

18.12. Create an error in the model (W)

18.35. For **Product1**: Change the definition of **p_Premium** to **a_Persons * a_WeeksX**.

18.13. Run all tests (W,WT)

18.36. Click on the **Test product model** icon. The test dialog appears.

18.37. Click on **Run all test cases**. The message **"Test case 'test1' returned '12' and now nothing. Disregard old value?"** appears.

18.38. Click **Cancel**.

18.14. Run test1 (W,WT)

18.39. Select **test1** from the **Test** combo box.

18.40. Click on **Compute**. The following message appears: **"Attribute/property 'a_WeeksX' not defined"**.

18.41. Click **Close**.

18.15. Position cursor to error (W)

18.42. In the **Toolbar**: Click on the icon **Position cursor to error** ( ). Note that the **Inspector** for **Product1** is opened (if not already opened) and that the cursor is positioned directly after the location of the error:

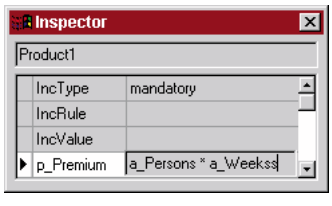


Figure 18.6. Cursor positioned directly after the location of the error

18.43. Delete the error.

18.44. Save the model (Zyx11.pms).

18.45. Generate a runtime model (overwrite previous version).

19. DBWin32 debugger messages (vpms.ini, W, DBWin32)

19.1. Enabling debug messages in VPMS.ini

19.1. Open **C:\Winnt\VPMS.ini**.

19.2. Add the following line (or edit the line if it already exists):

```
Debug=1
```

19.3. Save the file (Vpms12.ini).

19.2. Open DBWin32

19.4. Double-click on **C:\VPMS\wbench\Dbwin32.exe**. The DBWin32 main dialog appears.

Figure 19.1. DBWin32 main dialog

19.3. Test (W)

Workbench must be restarted after changes have been made in VPMS.ini.

19.5. Close the **Workbench**.

19.6. Double-click on **Zyx.pms**. The Workbench opens.

19.7. Open the **Test** dialog.

19.8. Select **test1**.

19.9. Click on **Compute**. Note that debugging messages now appear in DBWin32.

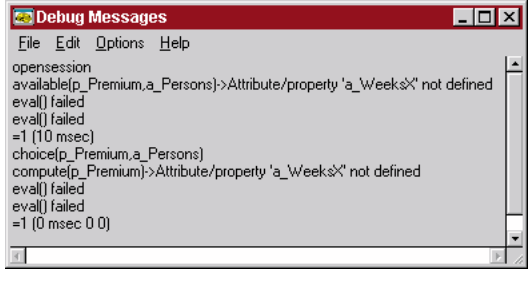


Figure 19.2. Debugging message in DBWin32

19.10. Close the **Test** dialog.

20. Testing the layout (.vpl, vpc) (D, DT)

Designer test is described in detail in the **Designer User's Guide**.

20.1. Test runtime layout (.vpc) within Designer (D,DT)

20.1. In the **Designer**: Click on the icon **Test application** ( ). A simplified **Designer Test** dialog is opened (the full dialog will be shown later).

20.2. Test the layout. Note that the results are the same as in the VFrame consultation.

20.3. Close the test dialog.

20.2. Modify layout; retest (D,DT)

20.4. Change the label **Insurance policy** to **Insurance policy 2**.

20.5. Click on the icon **Test application**. The messages appears: **Application was changed. Do you want to save it?**.

20.6. Click **Yes**. The test dialog with the change appears.

20.7. Close the test dialog.

20.3. Modify layout and save (D)

20.8. Change the label **Insurance policy** to **Insurance policy 3**.

20.9. Save the layout (do not test). Note that in directory **C:\VPL_Apps\Zyx** that the following files were updated when the layout was saved:

20.9.1. **Zyx.vpl**

20.9.2. **Zyx.vpc**

20.9.3. **Zyx.vpd**

20.4. Test layout outside Designer (DT)

20.10. Double-click on **Zyx.vpc**. The Designer Test dialog is opened on Zyx.vpc.

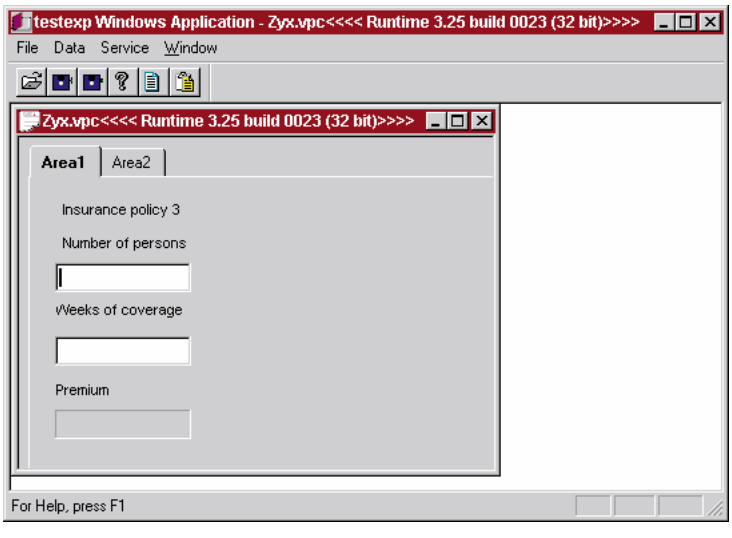


Figure 20.1. Designer Test dialog (outside Designer)

20.11. Test as within the Designer. Note that the results are the same.

20.12. Close the Designer test.

21. Testing the report with IFOS debug window (not available under NT ??)

21.1.

21.2.

22. Analyzing the report with Konzeptvorschau (RE, RV)

The Konzeptvorschau in the Report Viewer (opened from the Report Editor) provides an excellent view of information entered in the Report Editor for the report.

22.1. Open report (RE)

22.1. Double-click on **Zyx.cat**. The Report Editor opens.

22.2. Generate Konzeptvorschau (RE,RV)

22.2. In the **Report Editor**: From the main menu: Select **Datei / Konzeptvorschau...**. The Report Viewer opens with the Konzeptvorschau for the report.

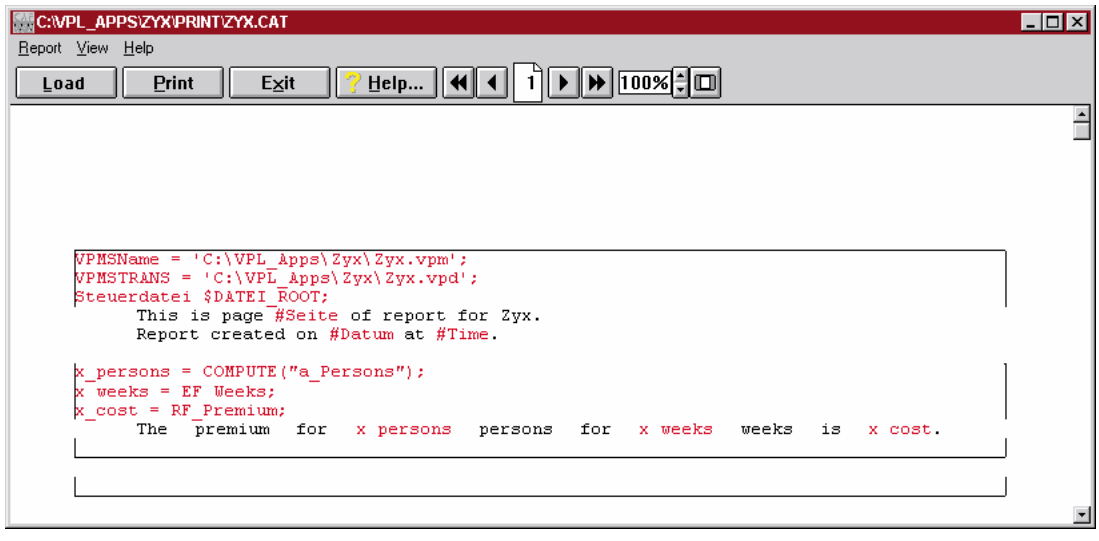


Figure 22.1. Konzeptvorschau for Zyx.cat in the Report Viewer

Note the bereichkommando information included for each bereich.

23. Testing in VFrame (VF) ??

23.1.
23.2.

24. (empty place-holder chapter)

24.1. .

25. Workareas (tabs) (D)

Each workarea in the layout has a tab to select it. When you create a layout, 2 workareas are created by default. Workareas can be added, deleted, and moved (the order of the workareas changed).

25.1. Popup menu for workarea (D)

25.1. Double-click on **Zyx.vpl** to open in the Designer.

25.2. Right-click on the tab **Area2**. The following pop-up dialog appears.

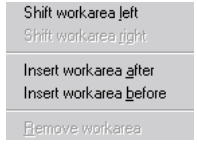


Figure 25.1. Popup dialog for workarea Area2

Note that the workarea can be moved to the left. Area2 cannot be removed because the layout must have a minimum of 2 workareas.

25.2. Inserting, removing, modifying properties (D)

25.3. Add a workarea.

25.4. Remove the added workarea.

25.5. Change the titles of the 2 workareas as shown in the following diagram:

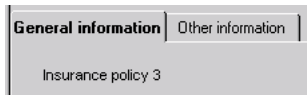


Figure 25.2. Workareas with new titles

25.6. Save the model (Zyx18.vpl).

26. Pages (in workarea frame) (D)

A workarea can contain any number of pages. A page can be displayed by clicking on the icon for the page in the frame on the left side of the workarea.

26.1. Set workarea style to Frame (D)

26.1. Select the workarea **Other information**.

26.2. In the **Inspector**: In tab **Workarea**: From the **Style** drop-down list: Select **Frame**. Note that a frame and a default page are created for the workarea:

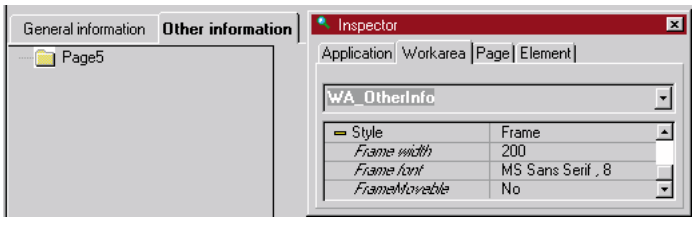


Figure 26.1. Page created in workarea Other information

26.2. Insert node (page) (D)

26.3. Right-click on the page icon. A pop-up dialog appears.

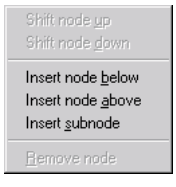


Figure 26.2. Pop-up dialog for page

26.4. Select **Insert node below**. Note that a second page (node) is added to the workarea (tab) (Zyx19.vpl).

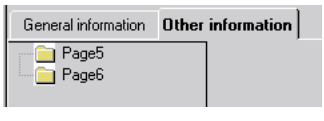


Figure 26.3. Second page (node) added to the workarea (tab)

27. Border, Line, Multimedia (D)

Borders (simple or 3d), lines, and multimedia elements (such as jpeg pics) can be added to the layout. The chapter provide a short demonstration of how to add such components.

27.1. Border (simple) (D)

27.1. In the **Designer**: Click on the icon **New border** ( ).

27.2. Click on the workarea **General information**. A border element is added.

27.3. Change the **Title** for the border element to **Border title text**.

27.4. Position the border as shown in the following text. Note that the border is displayed under the other elements.

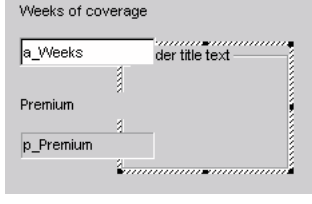


Figure 27.1. Border in the layout

27.2. 3d border (D)

27.5. Click on the icon **New 3d border** ( ).

27.6. Click on the workarea **General information**. A 3d border element is added.

27.7. Change the **Background color** to yellow.

27.8. Position the 3d border as shown in the following text. Note that the 3d border is displayed only above the border.

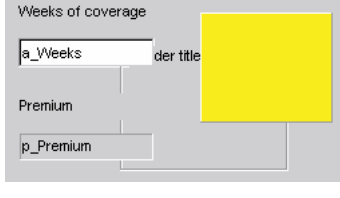


Figure 27.2. 3d border in the layout

27.3. Line (D)

27.9. Click on the icon **New line** ( ).

27.10. Click on the workarea **General information**. A line element is added (with default background color of blue).

27.11. Position the line as shown in the following text. Note that the line is displayed only above the border and 3d border (the line in this diagram has been resized).

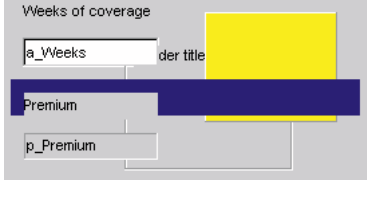


Figure 27.3. Line in the layout

27.4. Multimedia element (D)

27.12. Click on the icon **New multimedia element** ( ).

27.13. Click on the workarea **General information**. A mulitmedia element is added (with default image of VPMS).

27.14. For property **Media file** (under property **Type**): Select **c:\vpl_apps\zyx\print\zyx20.bmp**.

27.15. Position the multimedia element as shown in the following text. Note that the multimedia element is displayed only above the border, 3d border, and line. Note that the .bmp pic is not yet displayed.

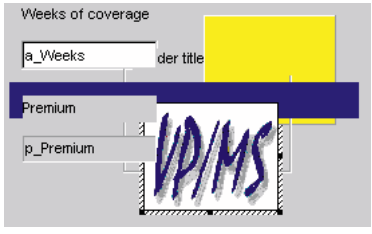


Figure 27.4. Multimedia (.bmp) element in the layout

27.16. Click on the **Test application** icon. Note that the .bmp is now displayed.

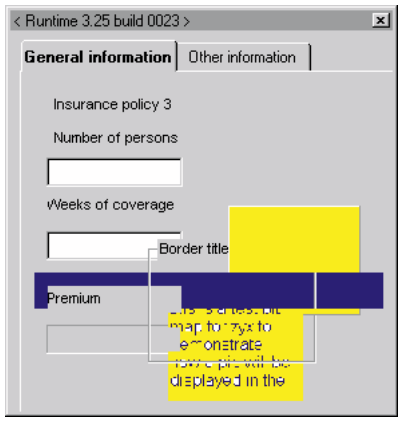


Figure 27.5. Multimedia element in the test layout

27.5. Modify the layout (D)

27.17. Rearrange and modify the layout elements as shown in the following diagram (note that the background color for label **Premium** has been changed to yellow also).

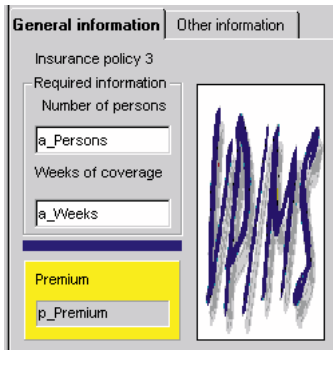


Figure 27.6. Modified layout with borders, line, and multimedia (.bmp) element

27.18. Click on the **Test application** icon. Note that the .bmp pic has been "stretched" to fit in the multimedia borders.

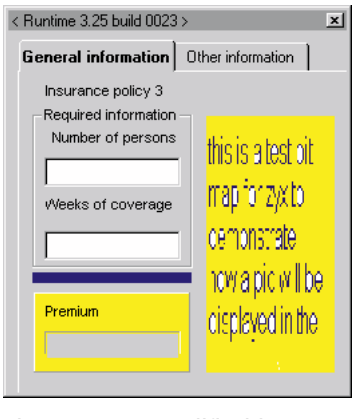


Figure 27.7. Modified layout with borders, line, and multimedia (.bmp) element in the test layout

27.19. Save the layout (Zyx20.vpl).

28. Report formatting (RE)

This chapter demonstrates how the following are created in the report:

- Header
- Footer
- Character formats
- Paragraph formats
- New pages
- Page layout

28.1. Report header (RE)

28.1. Open **Zyx.cat** in the **Report Editor**.

28.2. Add the following line to the beginning of the report:

```
Product report
```

28.3. Place the cursor anywhere in the above line.

28.4. From the main menu: Select **Bereiche / Kopfzeilen**.

28.2. Report footer (RE)

28.5. Select the following 2 lines:

```
This is page #Seite of report for Zyx.
Report created on #Datum at #Time.
```

28.6. From the main menu: Select **Bereiche / Fusszeilen**.

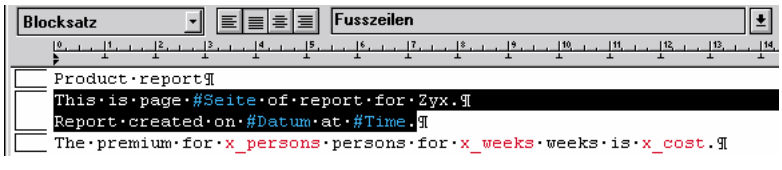


Figure 28.1. Header and footer in report

28.3. Create new character style (RE)

NOTE: RE does not allow changing the standard character format for a paragraph format. Thus, in order to change the format of characters in a paragraph, the character format must be changed (as demonstrated in this section).

28.7. Select the following line (the header text):

```
Product report.
```

28.8. Select **Format / Zeichen**. The **FormatAuswahl** dialog appears:

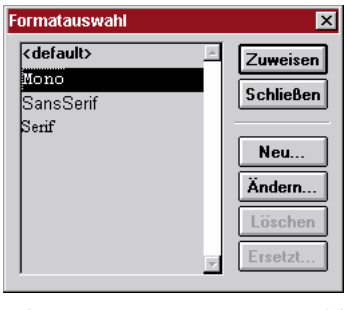


Figure 28.2. FormatAuswahl dialog

28.9. Click **Neu...**. The standard **Schriftart** dialog appears:

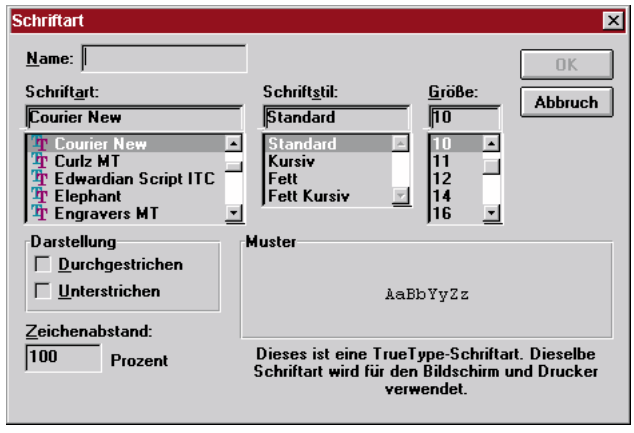


Figure 28.3. Standard Schriftart dialog

28.10. Select **Schriftstil Fett**.

28.11. Select **Grosse 14**.

28.12. In the **Name:** entry field: Enter **Header**.

28.13. Click **OK**. The **Formatauswahl** dialog regains the focus.

28.14. Click **Zuweisen**. Note the new text style.

Figure 28.4. New Header style text

28.4. Create new paragraph style (RE)

28.15. Add the following text

```
Premium total
```

before the following line (and in the same bereich):

```
The premium for x_persons persons for x_weeks weeks is x_cost.
```

28.16. Place the cursor within the following text:

```
Premium total
```

28.17. Select **Format / Absatz**. The **Formatauswahl** dialog appears.

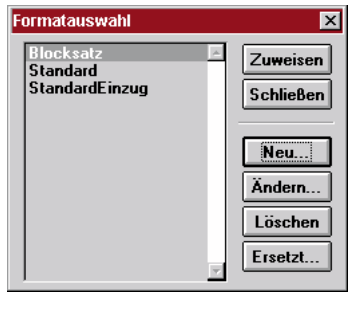


Figure 28.5. Formatauswahl dialog

28.18. Click on **Neu...**. The **Absatzformat** dialog appears:

Figure 28.6. Absatzformat dialog

28.19. Select **Rahmen**. The **Absatzrahmen** dialog appears.

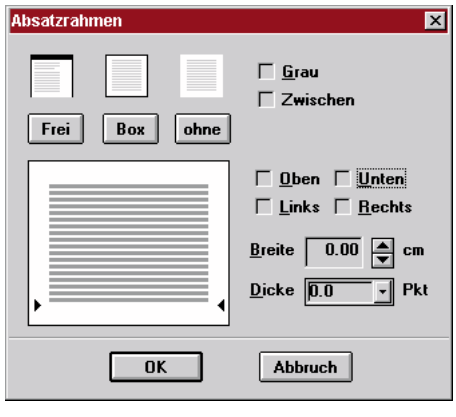


Figure 28.7. Absatzrahmen dialog

28.20. Click the checkbox **Unten**.

28.21. From the **Dicke** combo box: Select **1**.

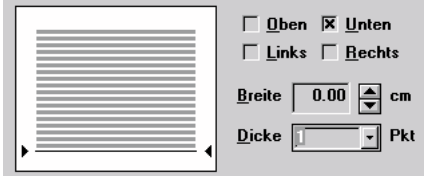


Figure 28.8. Selecting the box lines and thickness

28.22. Select **OK**. The **Absatzformat** dialog regains the focus.

28.23. Change **Zeilenabstände / Unten** to **0.20 cm**.

28.24. In the **Name** entry field: Enter **ZyxHeading1**.

28.25. Select **OK**. The **Formatauswahl** dialog regains the focus.

28.26. Select **Zuweisen**. Note the new format.

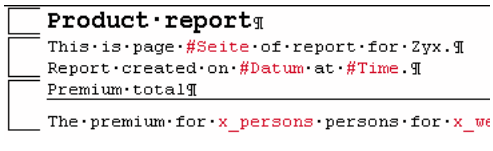


Figure 28.9. ZyxHeading1 format

28.27. Click on the **Paragraph formats** combo box. Note that the new format is included.

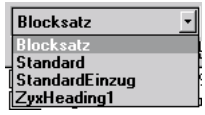


Figure 28.10. ZyxHeading1 in the paragraph formats combo box

28.5. Create new paragraph style for header (RE)

28.28. Create a new paragraph style **Header** with **Zeilenabstand** = **0.5 cm**.

28.29. Set the paragraph style for the first line to **Header**.

Figure 28.11. Paragraph style Header

28.6. Insert page break (RE)

28.30. Add the following text at the end of the report:

```
The last page.
```

28.31. Place the cursor at the beginning of the line.

28.32. Press **CTRL-RET**. Note that a page break has been inserted.

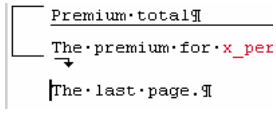


Figure 28.12. Page break

28.7. Modify page layout (RE)

28.33. Select **Format / Dokument**.

28.34. For **Seitengroesse**: Set **Breite** to **13**.

28.35. For **Seitengroesse**: Set **Hoehe** to **7**.

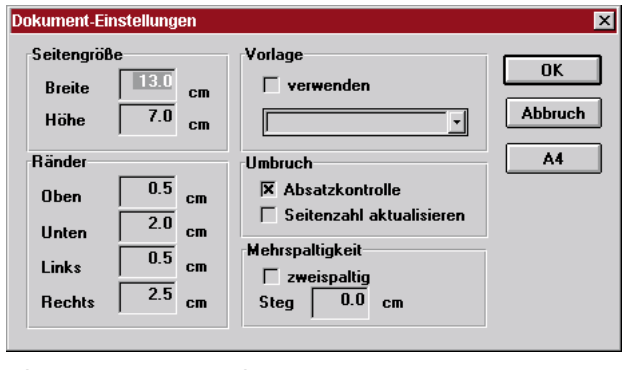


Figure 28.13. Page layout

28.36. Click **OK**.

28.37. Save the report (Zyx21.cat).

28.8. Test (VF)

28.38. Open the **Zyx consultation** in VFrame.

28.39. Enter for **Persons**: **3**.

28.40. Enter for **Weeks**: **4**.

28.41. Print the report. The following pages are created:

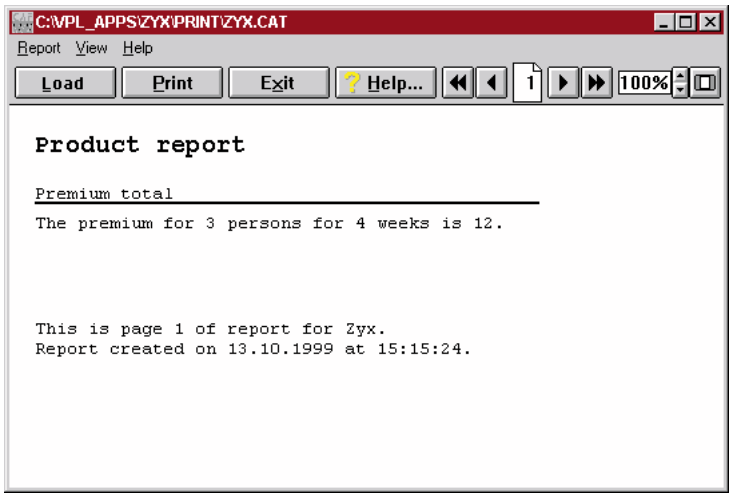


Figure 28.14. Product report printout page 1

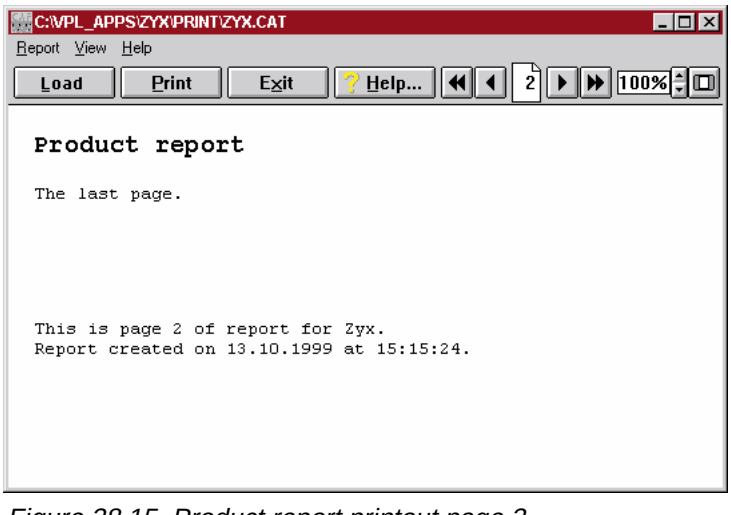


Figure 28.15. Product report printout page 2

29. Specifying default values for attributes (W)

29.1. Add property "default" for a_Persons, a_Weeks (W)

29.1. For attribute **a_Persons**: Add property **default** with value **5**.

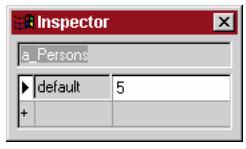


Figure 29.1. a_Persons property default

29.2. For attribute **a_Weeks**: Add property **default** with value **2**.

29.3. Save the model (Zyx22.pms).

29.4. Generate a runtime model.

29.2. Test (VF)

29.5. Restart **VFrame** (when a change has been made to the runtime model, VFrame must be restarted).

29.6. Select the Zyx consultation. Note that the default values for **a_Persons** and **a_Weeks** are displayed.

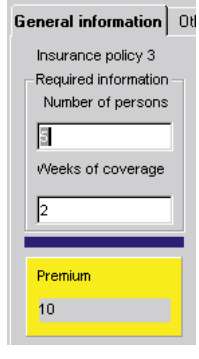


Figure 29.2. Default values for a_Persons, a_Weeks

30. Check value assigned to attribute (W)

30.1. Add property check for a_Persons; create runtime (W)

30.1. For attribute **a_Persons**: Add property **check**.

30.2. Double-click in the table cell to the right of the property **check** (column 2). The **Editor** appears.

30.3. Enter the following code in the Editor window:

```
if (a_Persons<1;
  error("Specify at least 1 person";"a_Persons");
  if (a_Persons>5;
    error("5 persons max";"a_Persons");a_Persons))
```

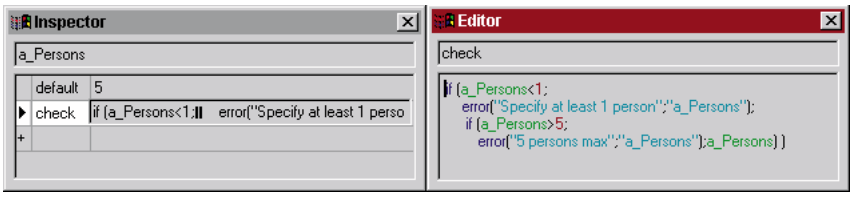


Figure 30.1. Check code for a_Persons in the Editor window

Note: If the text colors do not match those in the above diagram: A typing error has probably occurred.

30.4. Save the model (Zyx23.pms).

30.5. Create a runtime model.

30.2. Test (W,WT)

30.6. Click on the **Test product model** icon.

30.7. Select **test1**.

30.8. Enter **6** for **a_Persons**.

30.9. Click **Compute**. Note the error message in the Test window:

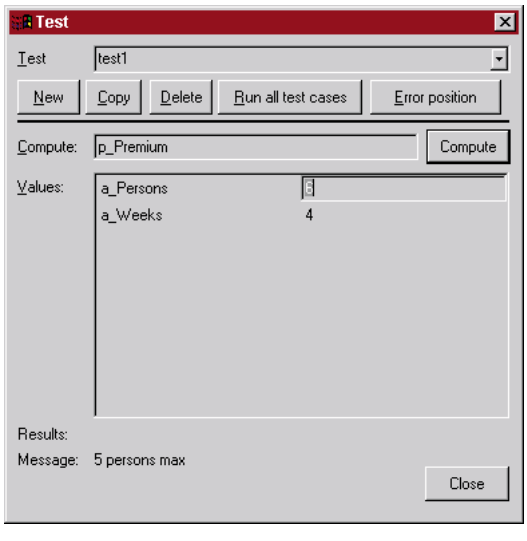


Figure 30.2. Error message in Test window

30.10. Close the test window.

30.3. Test (VF)

30.11. Restart **VFrame**.

30.12. Select the Zyx consultation.

30.13. Enter for **a_Persons**: **6**.

30.14. Click **tab** to change the focus. Note the message in the VFrame status bar.

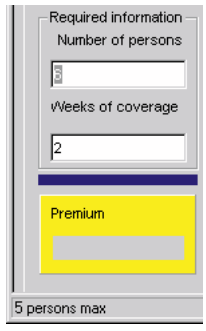


Figure 30.3. VFrame status bar message for 5 person max

31. Data indicator (W,D)

A data indicator is a visual indication that data is required on a workarea or page.

31.1. Create data indicator property (W)

31.1. For **Zyx.pms**: For **Product1**: Add property **DI_Product1** with the following definition:

```
if(
   not(undefined("a_Persons"))
   &&
   ( (a_Persons>0) && (a_Persons<6) )
   &&
   not(undefined("a_Weeks"))
   &&
   (a_Weeks>0);
   1;
   0)
```

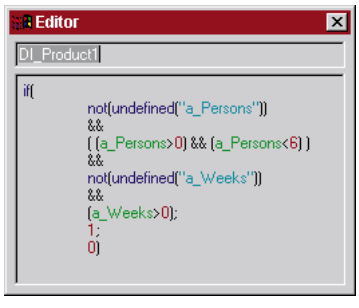


Figure 31.1. DI_Product1 definition

31.2. Save the model (Zyx24.pms).

31.3. Create a runtime model.

31.2. Specify indicator for layout (D)

31.4. For workarea **General information**: Set **Data indicator** to **On**.

31.5. Set **DI Rule** to **DI_Product1**.

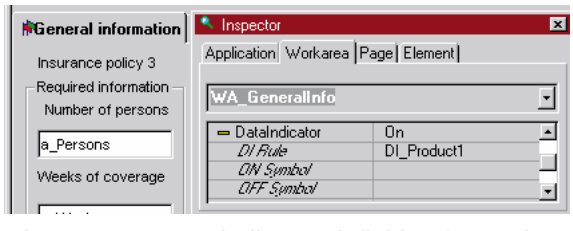


Figure 31.2. Data indicator definition for workarea General information in layout

31.6. Save the layout (Zyx24.vpl).

31.3. Test (D,DT)

31.7. In the **Designer**: Click on the **Test** icon. Note the **positive** data indicator on the workarea tab (the default values are valid):

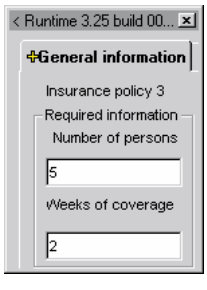


Figure 31.3. Positive data indicator on General information tab

31.8. Change **Number of persons** to **6**.

31.9. Click to change the focus. Note the **negative** data indicator on the workarea tab:

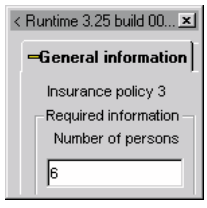


Figure 31.4. Negative data indicator on General information tab

32. Use Combo box / radio button group to select attribute values (W,D,RE)

32.1. Add property table; create table of possible values; create runtime (W)

32.1. For attribute **a_Weeks**: Add property **table** with value **"t_Weeks"** (include the quotation marks).
32.2. Create table **t_Weeks**.
32.3. For table **t_Weeks**: Add key **1** with value **1 week** (no quotations marks).
32.4. For table **t_Weeks**: Add key **2** with value **2 weeks** (no quotations marks).
32.5. Save the model (Zyx25.pms).
32.6. Create a runtime model.

32.2. Test (W,WT)

32.7. Click on the **Test product model** icon.
32.8. Select **test1**.
32.9. Click on the entry field for **a_Weeks**. A **Combo box** for a_Weeks with default value of **2 weeks** appears:

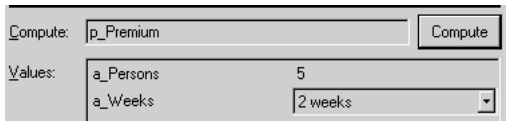


Figure 32.1. a_Weeks in combo box in the Workbench test dialog

32.10. Click **Compute**. The result of **6** is displayed.
32.11. Close the **Test** dialog.

32.3. Replace edit field with combo box for a_Weeks (D)

32.12. In the **Designer**: Select the **edit field** for **a_Weeks**.
32.13. In the **Inspector**: Click on the entry field to the right of **Type** ("Text"). A combo box appears with the available types of components.
32.14. Select **Combo box**.

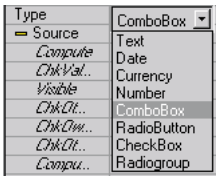


Figure 32.2. Selecting Combo box for a_Weeks

32.15. Rename the Combo box element to **CB_Weeks**.

32.4. Add radio button group for selection of a_Weeks (D)

32.16. Click on the **New radiogroup** button () in the toolbar.
32.17. Click with the cursor on the layout. A radiogroup is added to the layout.
32.18. Set the element name to **RG_Weeks**.
32.19. Set the radio group **Source** to **a_Weeks**.

32.20. Set **Compute** to **OnChange**.

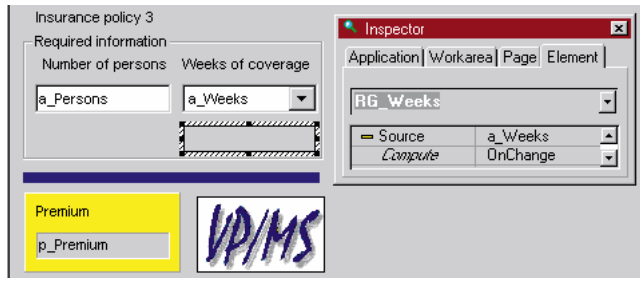


Figure 32.3. Radio group and settings for a_Weeks in the layout

32.21. Save the layout (Zyx25.vpl).

32.5. Test (D,DT)

32.22. Click the **Test Application** button in the toolbar. Note that the dialog appears WITHOUT the radio button group. This is because the runtime model has changed since Designer was last started.

32.23. Close the Designer.

32.24. Open **Zyx.VPL** in Designer.

32.25. Click the **Test Application** button in the toolbar. The following dialog appears.

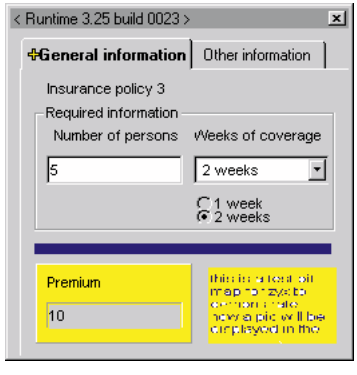


Figure 32.4. Radio group for a_Weeks in the test layout

32.26. Select **1 week** from the combo box. Note that the Premium is not immediately calculated (Compute on Exit).

32.27. Click in the **Persons** entry field. Note that the Premium is recalculated.

32.28. Select **2 weeks** from the **Radio Group**. Note that the Premium is immediately calculated (Compute on Change).

32.29. Close the Designer test dialog.

32.6. Modify report for changed element name (RE)

32.30. Change the following line in the bereichskommando

```
x_weeks = EF_Weeks;
```

to

```
x_weeks = CB_Weeks;
```

32.31. Save the report.

32.7. Test (VF)

32.32. Restart **VFrame**.

32.33. Test the radio group in VFrame. Note that the results are the same as for the test in the Designer.

32.34. Print the report. Note that the output for the report has changed:

Product report

Premium total

The premium for 5 persons for 2#2 weeks weeks is 10.

Figure 32.5. Report with a_Weeks from Combo box (error)

32.8. Modify report for text input from table (RE)

The key and the text in the table are returned. This must be filtered.

32.35. In the bereichskommando: Replace the following

```
x_weeks = CB_Weeks;
```

with

```
i=instr(CB_Weeks,"#");
x_weeks = mid$(CB_Weeks,i+1,8);
```

32.36. Delete the word **weeks** in the following text

```
The premium for x_persons persons for x_weeks weeks is x_cost.
```

32.37. Save the report (Zyx25.cat).

32.9. Test (VF)

32.38. Print the report (VFrame does not need to be restarted). Note that the printout is now correct.

33. Select attribute value from multiple 2-column tables (W,D,RE)

33.1. Add attribute a_CoverageLevel selected from combo box (W)

33.1. Add attribute **a_CoverageLevel** with property **table** with value **"t_CoverageLevel"** (include the quotation marks).

33.2. Add table **t_CoverageLevel** with the following keys / values:

33.2.1. **1** / **1 K**

33.2.2. **2** / **10 K**

33.2.3. **3** / **100 K**

33.3. For **a_CoverageLevel**: Add property **default** with value **1**.

33.2. Create coverage premium tables for 1 week, 2 weeks (W)

33.4. Create table **t_CoveragePremium1W** with the follwing keys / values:

33.4.1. **1** / **1**

33.4.2. **2** / **8**

33.4.3. **3** / **60**

33.5. Create table **t_CoveragePremium2W** with the follwing keys / values:

33.5.1. **1** / **0.8**

33.5.2. **2** / **6.4**

33.5.3. **3** / **48**

33.3. Add Product1 property p_CoveragePremium (W)

33.6. For **Product1**: Add property **p_CoveragePremium** with the following code:

```
if ( a_Weeks = 1 ;
   t_CoveragePremium1W [ a_CoverageLevel ] ;
   t_CoveragePremium2W [ a_CoverageLevel ] )
```

33.4. Edit calculation of p_Premium (W)

33.7. For Product **Product1**: Change the code for property **p_Premium** to the following:

```
a_Persons * a_Weeks * p_CoveragePremium
```

33.8. Save the model (Zyx26.pms).

33.9. Create a runtime model.

33.5. Test (W,WT)

33.10. Click on the **Test product model** icon.

33.11. Select **test1**.

33.12. Click **Compute**. A combo box box for **a_CoverageLevel** appears:

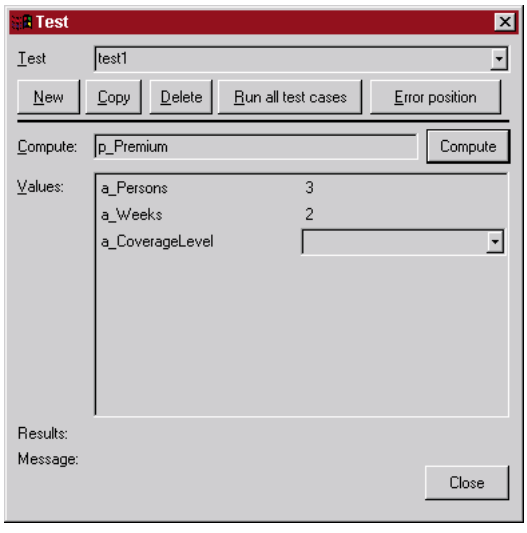


Figure 33.1. Test window with a_CoverageLevel combo box

33.13. Select **100 K**.

33.14. Click **Compute**. Note the result: 288 = 3 * 2 * 48 (coverage level premium from t_CoveragePremium2W for 100 K coverage)

33.15. Close the Test dialog.

33.6. Add radio button group in layout (D)

33.16. In the **Designer**: Add a radio group named **RG_CoverageLevel**.

33.17. Attempt to specify **a_CoverageLevel** as the source for the radio group. Note that a_CoverageLevel is not in the list of attributes.

33.18. From the main menu: Select **Options / Workbench synchronisation**.

33.19. Specify the **Source** for the radio button group as **a_CoverageLevel**.

33.20. Select **Compute On Change**.

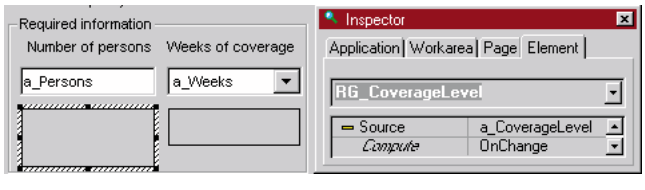


Figure 33.2. Radio button group for a_CoverageLevel in Designer

33.21. Save the layout (Zyx26.vpl).

33.7. Test (D,DT)

33.22. In the toolbar: Click on the **Test Application** icon. The following test dialog appears.

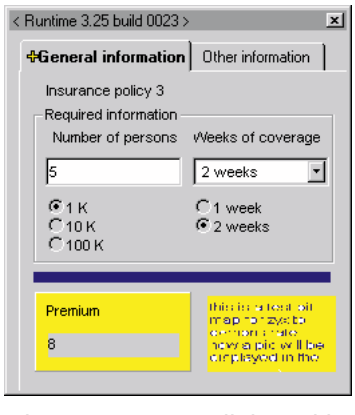


Figure 33.3. Test dialog with a_CoverageLevel radio buttons

33.23. Close the test dialog.

33.8. Add to report (RE)

33.24. Add the text below in bold:

```
The premium for x_persons persons for x_weeks with coverage level
x_coveragelevel is x_cost.
```

33.25. Specify "x_coveragelevel" as a data field.

33.26. Add the following line to the bereichskommando:

```
i=instr(RG_CoverageLevel,"#");
x_coveragelevel = mid$(RG_CoverageLevel,i+1,6);
```

33.27. Save the report (Zyx26.cat).

33.9. Test (VF, RV)

33.28. Restart **VFrame**.

33.29. Open the consultation.

33.30. Print a report. Note that the printed value for the coverage level:

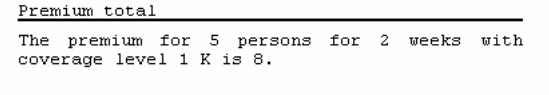


Figure 33.4. Printout of coverage level

33.31. Close the report.

34. Select attribute value from single 3+-column table (W)

34.1. Create table t_CoveragePremium (W)

34.1. Create table **t_CoveragePremium**.
34.2. Double-click on the table.
34.3. From the main menu: Select **Edit / Add column**. Note the new column in the table.
34.4. Change the name of the second (middle) column to **weeks1**.
34.5. Change the name of the third (right) column to **weeks2**.
34.6. Add the following to the colums key / weeks1 / weeks2 in the table:
34.6.1. **1** / **1** / **0.8**
34.6.2. **2** / **8** / **6.4**
34.6.3. **3** / **60** / **48**

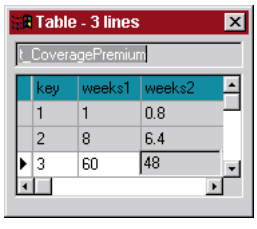


Figure 34.1. t_CoveragePremium table in the Workbench

34.2. Modify p_CoveragePremium (W)

34.7. Replace the **p_CoveragePremium** code with the following:

```
if ( a_Weeks = 1 ;
   t_CoveragePremium[a_CoverageLevel].weeks1 ;
   t_CoveragePremium[a_CoverageLevel].weeks2 )
```

34.8. Save the model (Zyx27.pms).
34.9. Generate a runtime model.

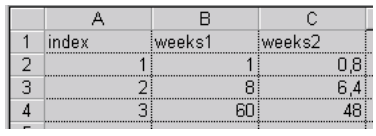
34.3. Test (WT, DT, VF, RV)

34.10. Test as in the previous chapter. Note that the functionality has not changed.

35. Import a table from Excel database (Excel,W)

35.1. Create Excel table (Excel)

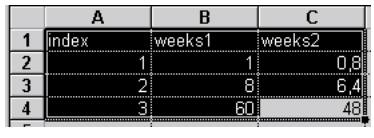
35.1. Create the following Excel table:



	A	B	C
1	index	weeks1	weeks2
2	1	1	0,8
3	2	8	6,4
4	3	60	48

Figure 35.1. Excel table for import

35.2. Select the colums A..C and rows 1..4.



	A	B	C
1	index	weeks1	weeks2
2	1	1	0,8
3	2	8	6,4
4	3	60	48

Figure 35.2. Selected columns and rows for import in the Excel table

35.3. From the Excel main menu: Select **Insert / Name / Define**. The **Define name** dialog appears.
35.4. Enter **Zyx**.
35.5. Click **Add**.
35.6. Click **OK**. Note the name of the selected cells in the Excel main dialog:

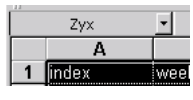


Figure 35.3. Name of the selected cells in the Excel main dialog

35.7. Save the database as **C:\VPL_Apps\Data_Bas\Zyx.xls** (Zyx28.xls). Do not close **Excel**.

35.2. Import table (W)

35.8. In the **Workbench**: Right-click on **t_CoveragePremium**. A popup menu appears.
35.9. Select **Import data file**. The **Table import** dialog appears.
35.10. Enter the following SQL statement in the dialog:

```
select * from Zyx
```

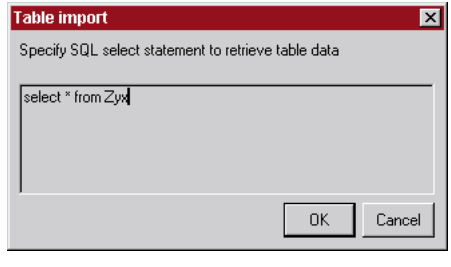


Figure 35.4. SQL code for importing table

35.11. Click **OK**. The **Select Database** dialog appears.
35.12. Select tab **Computer data sources**:

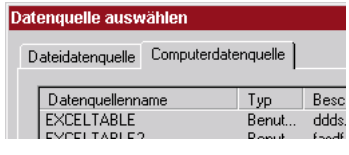


Figure 35.5. Tab Computer data sources

35.13. Select **New**. The **Create new datasource** dialog appears.

35.14. Select **User datasource**.

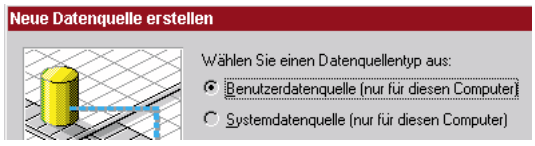


Figure 35.6. Create new datasource dialog

35.15. Select **Next** (create user datasource for this computer).
35.16. Select the **Microsoft Excel driver (.xls)**.
35.17. Select **Next**.
35.18. Select **Ready**. The **ODBC Microsoft Excel setup** dialog appears.

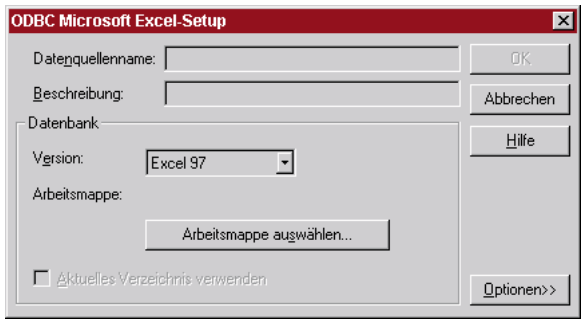


Figure 35.7. Excel setup dialog

35.19. Click **Select Workmap...**.
35.20. Double-click on **C:\VPL_Apps\Data_Bas\Zyx.xls**.
35.21. Enter **Zyx** as the database name.

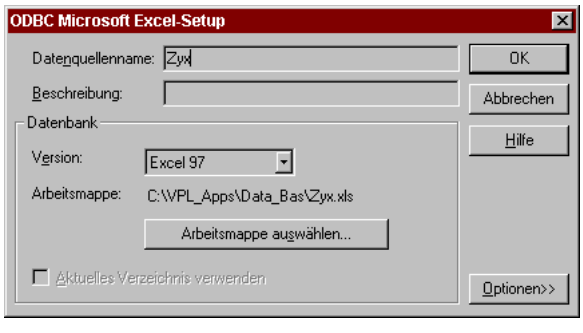


Figure 35.8. Excel setup dialog with required information

35.22. Select **OK**. The **Select datasource** dialog regains the focus.
35.23. Select **OK**. An **Inspector** appears in the workbench for the table with the imported database data.

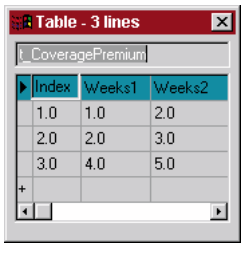


Figure 35.9. Inspector for table in Workbench with imported database data

35.24. CHANGE DECIMAL NUMBERS IN FIRST COLUMN TO INTEGERS. Note that the table is the same as previously.
35.25. Save the model (Zyx28.pms).
35.26. Create a runtime model.

36. Table contents direct from dBase file (Excel,W)

The contents of a dBase file can be read dynamically.

36.1. Save modified Zyx.xls as dBase 4 file Zyx.dbf (Excel)

36.1. In **Excel**: Change the value of **48** to **50**.
36.2. From the main menu: Select **File / Save as...**.
36.3. Save the file in **DBF 4** format to file **C:\VPL_Apps\Zyx\Zyx.dbf**.
36.4. A message appears about DB4 not supporting maps. Click **OK**.

36.2. Modify t_CoveragePremium for the DB4 file (W)

36.5. Place the cursor in the last column of **t_CoveragePremium**.
36.6. From the main menu: Select **Edit / Remove column**.
36.7. Rename **column 1** to $**DBASE**.
36.8. Rename **column 2** to **Langtext**.
36.9. Delete all remaining rows.
36.10. Add **$DBASE / Langtext** row with values **file / ZYX.DBF**.
36.11. Add **$DBASE / Langtext** row with values **oem / 1**.

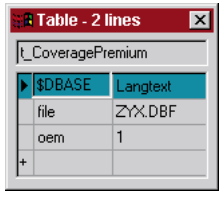


Figure 36.1. DBF 4 definition in table t_CoveragePremium

36.12. Save the model (Zyx29.pms).
36.13. Create a runtime model.

36.3. Test (W,WT)

36.14. Select **test1**.
36.15. Select **2 weeks**.
36.16. Select **100 K**. Note the value of **300** reflects the 50 in the .dbf file.
36.17. Close the test dialog.

36.4. Modify value in .dbf file (Excel)

36.18. In **Excel**: Change the value of **50** to **51**.
36.19. From the main menu: Select **File / Save as...**.
36.20. Save the file in **DBF 4** format to file **C:\VPL_Apps\Zyx\Zyx.dbf**.
36.21. A message appears about DB4 not supporting maps. Click **OK** (Zyx29.dbf).

36.5. Test (W,WT)

36.22. Select **test1**.
36.23. Select **2 weeks**.
36.24. Select **100 K**. Note the value of **306** reflects the 51 in the .dbf file.
36.25. Close the test dialog.

37. Dynamic calculation of default (W,D)

A2.default will be dynamically recalculated if A2 is included in the property **A1.dynamic** in the model and **A1.ChkDynamic** is set to **Yes** in the layout.

37.1. Add a_Payment to model (W)

37.1. Add attribute **a_Payment** with property **default** with value **5**.

37.2. Add a_CoverageLevel property dynamic (W)

37.2. Add **a_CoverageLevel** property **dynamic** with value **"a_Payment"** (include the quotation marks).

37.3. Save the model (Zyx30.pms).

37.4. Create a runtime model.

37.3. Add label and entry field for a_Payment (D)

37.5. Add a **label** with **Name** as **Label5**.

37.6. Set the **Title** to **Payment**.

37.7. Add an **Entry field** with **Name** as **EF_Payment**.

37.8. Set the **Source** to **a_Payment**.

37.4. Enable dynamic checking for a_CoverageLevel radio group (D)

37.9. For the **a_CoverageLevel** radio group: Set property **ChkDynamic** to **Yes**.

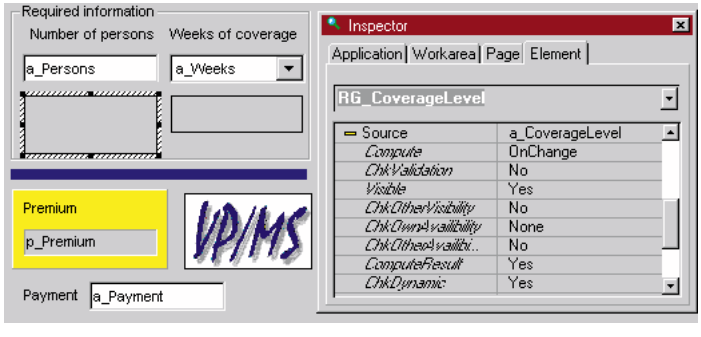


Figure 37.1. Dynamic checking settings in the layout

37.10. Save the layout (Zyx30.vpl).

37.5. Test (D,DT)

37.11. Click on the **Test layout** icon. The following dialog appears.

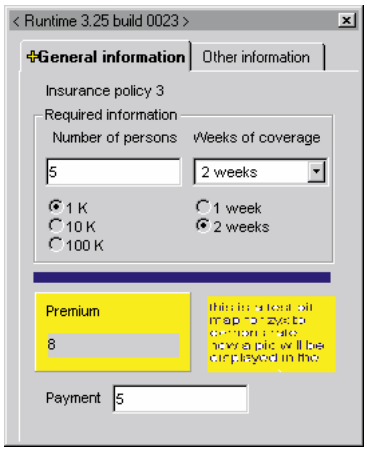


Figure 37.2. Test layout with dynamic checking

37.12. Enter **0** for **Payment**.

37.13. Select **1 week**. Note that the Payment doesnt change.

37.14. Enter **3** for **Number of persons**.

37.15. Click in the **Payment** entry field. Note that the Payment doesnt change.
37.16. Select **10 K**. The **Payment** field changes back to **5**.
37.17. Close the test dialog.

38. Dynamic table contents (filter) (W,D)

The accessible contents of a table can be controlled with the **filter** property.

38.1. Add a_CoverageLevel property filter (W)

38.1. Add **a_CoverageLevel** property **filter** with definition:

```
if (a_Weeks=2;key>1;key>0)
```

38.2. Add a_Weeks property dynamic (W)

38.2. Add **a_Weeks** property **dynamic** with value **"a_CoverageLevel"** (include the quotation marks).

38.3. Save the model (Zyx31.pms).

38.4. Create a runtime model.

38.3. Enable dynamic checking for a_Weeks radio group (D)

38.5. For the **a_Weeks** radio group: Set property **ChkDynamic** to **Yes**.

38.6. Save the layout (Zyx31.vpl).

38.4. Test (D,DT)

38.7. Click on the **Test layout** icon. The following dialog appears.

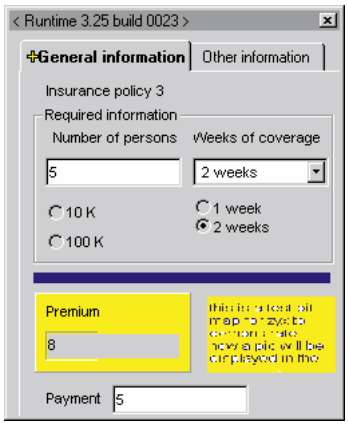


Figure 38.1. Test layout with filtered table content

38.8. Select radio button **1 week**. Note that the radio button for **1 K** is now available.

Figure 38.2. 1K radio button available after 1 week selected

38.9. Close the test dialog.

39. Dynamic label (W,D)

The label text can be coupled to an attribute and change dynamically.

39.1. Add a_Payment property label (W)

39.1. Add **a_Payment** property **label** with definition:

```
if (p_Premium>100;"Payment (required)";"Payment (optional)")
```

39.2. Save the model (Zyx32.pms).

39.3. Create a runtime model.

39.2. Add Payment label property Attribute (D)

39.4. For the **Payment** label: Set property **Attribute** to **a_Payment**.

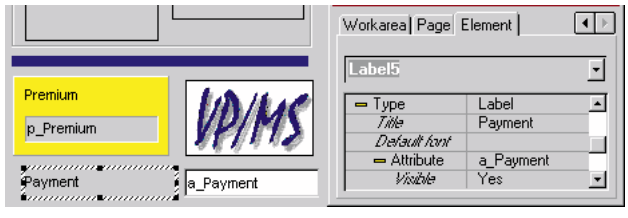


Figure 39.1. Attribute settings for Payment label

39.5. Save the layout (Zyx32.vpl).

39.3. Test (D,DT)

39.6. Click on the **Test layout** icon. Note the new label.

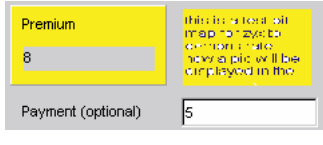


Figure 39.2. Test layout with dynamic label

39.7. Select radio button **100 K**. Note that the **Premium > 100** and the label has changed accordingly

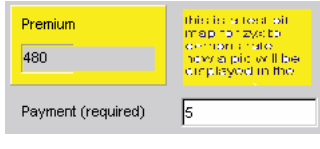


Figure 39.3. Test layout dynamic label with Premium > 100

39.8. Close the test dialog.

40. Dynamic availability (W,D)

The availability of an entry field can be dynamically enabled / disabled.

40.1. Add Product1 property p_PaymentAvailable (W)

40.1. For **Product1**: Add property **p_PaymentAvailable** with definition:

```
if ( p_Premium < 400; a_Payment; 12345)
```

40.2. Save the model (Zyx33.pms).

40.3. Create a runtime model.

40.2. Add availability list for a_Payment entry field (D)

40.4. For the **a_Payment** entry field: Set property **ChkOwnAvailability** to **List**.

40.5. Set property **List (...;...)** to **p_PaymentAvailable**.

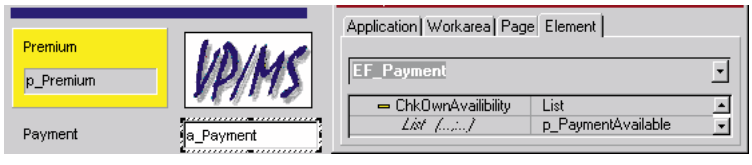


Figure 40.1. Availability settings for Payment entry field

40.6. For the **a_CoverageLevel** radio group: Set property **ChkOtherAvailability** to **Yes**.

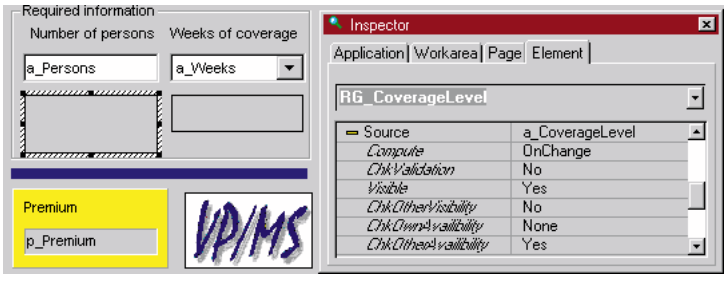


Figure 40.2. Availability settings for CoverageLevel radio group

40.7. Save the layout (Zyx33.vpl).

40.3. Test (D,DT)

40.8. Click on the **Test layout** icon. Note that the payment field can be editted, since the Premium < 400.

40.9. Select **100 K**. Now Premium = 480. Note that the Payment entry field is not available (gray with no input accepted).

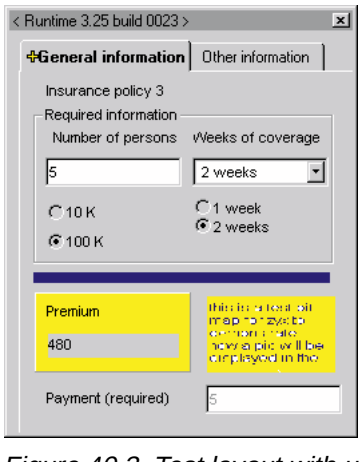


Figure 40.3. Test layout with unavailable Payment entry field

40.10. Close the test dialog.

41. Dynamic visibility (W,D)

The visibility elements can be dynamically enabled / disabled.

41.1. Add a_Payment property visible (W)

41.1. For **a_Payment**: Add property **visible** with definition:

```
if (p_Premium>100; 1; 0)
```

41.2. Save the model (Zyx34.pms).

41.3. Create a runtime model.

41.2. Add visible property for a_Payment entry field (D)

41.4. For the **a_Payment** entry field: Set property **Visible** to **Value**.

41.5. Set property **Property** to **a_Payment.visible**.

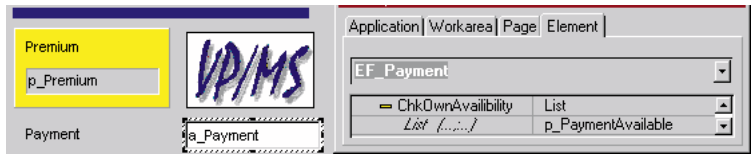


Figure 41.1. Visibility settings for Payment entry field

41.3. Add visible property for a_Payment entry field (D)

41.6. For the **Payment** label: Set property **Visible** to **Check**.

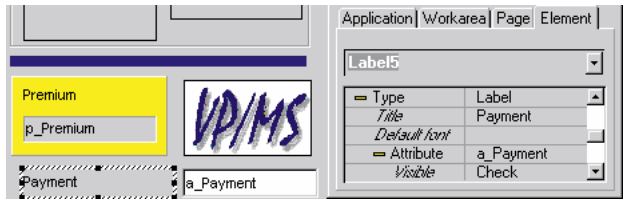


Figure 41.2. Visibility settings for Payment label

41.4. Specify ChkOtherVisibility for a_CoverageLevel radio group (D)

41.7. For the **a_CoverageLevel** radio group: Set property **ChkOtherVisibility** to **Yes**.

41.8. Save the layout (Zyx34.vpl).

41.5. Test (D,DT)

41.9. Click on the **Test layout** icon. Note that the payment label/field are not displayed, since Premium < 100.

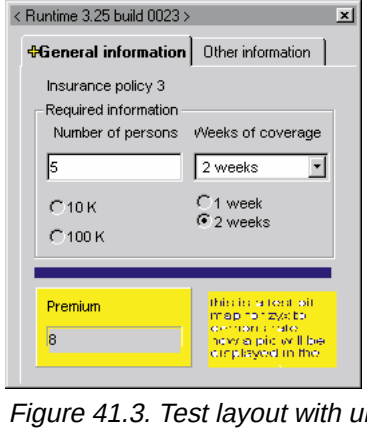


Figure 41.3. Test layout with unvisible Payment lable/entry field

41.10. Select **100 K**. Now Premium = 480. Note that the Payment label/entry are displayed.

41.11. Enter **1** for **Number of persons**.

41.12. Select **1 week**. Note that although the **Premium** is **1**, the Payment label/entry fields are still visible.

This is because ChkOtherVisibility was set only for the Coverage Level radio group.

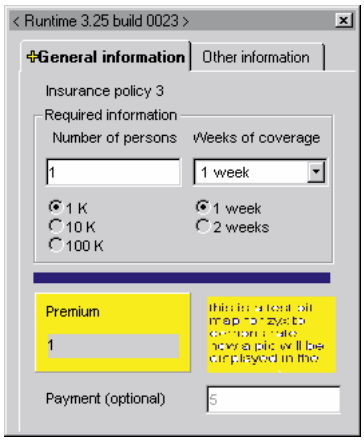


Figure 41.4. Test layout with visible Payment label / entry field

41.13. Select **10 K**. The Payment label/entry fields are not visible. This is because ChkOtherVisibility was set for the Coverage Level radio group.

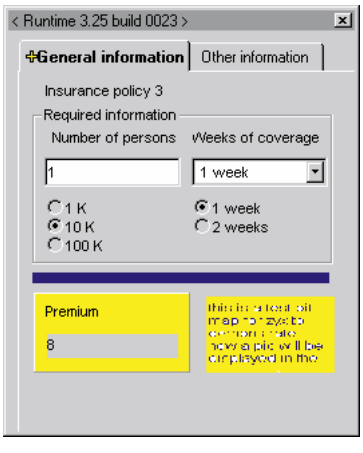


Figure 41.5. Test layout with visible Payment label / entry field

41.14. Close the test dialog.

42. Dynamic layout elements in the report (RE)

Elements that are not visible in the layout should not be visible in the report.

42.1. Add text to Premium total bereich (RE)

42.1. Add the following line to the report in front of the manual page break.

```
x_paymentlabel is x_payment.
```

42.2. Select the following text

```
Premium total
The premium for x_persons persons for x_weeks with coverage level
x_coveragelevel is x_cost.
x_paymentlabel is x_payment.
```

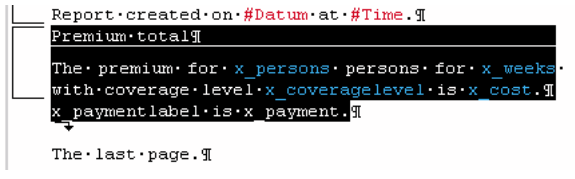


Figure 42.1. Selected text for the bereich

42.3. Select **Ansicht / Persistente Selektion**.

42.4. Click anywhere in the report. Note that the selected text stays selected.

42.5. Click within the **Premium total** bereich. Note the bereichskommando text in the bereich.

42.6. Select **Bereich / Setzen**. Note that the bereich has been extended to included all marked text.

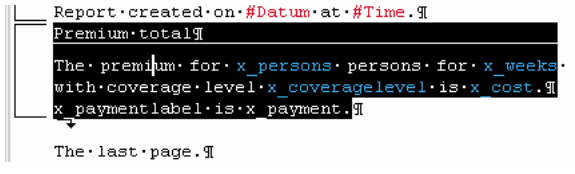


Figure 42.2. Extended bereich

42.2. Create subbereich for added text (RE)

42.7. Deselect **Ansicht / Persistente Selektion**.

42.8. Click anywhere in the report. Note that the bereich is no longer selected.

42.9. Select the added text:

```
x_paymentlabel is x_payment.
```

42.10. Select **Bereich / Neu**. The Bereichskommando editor is opened for the bereich.

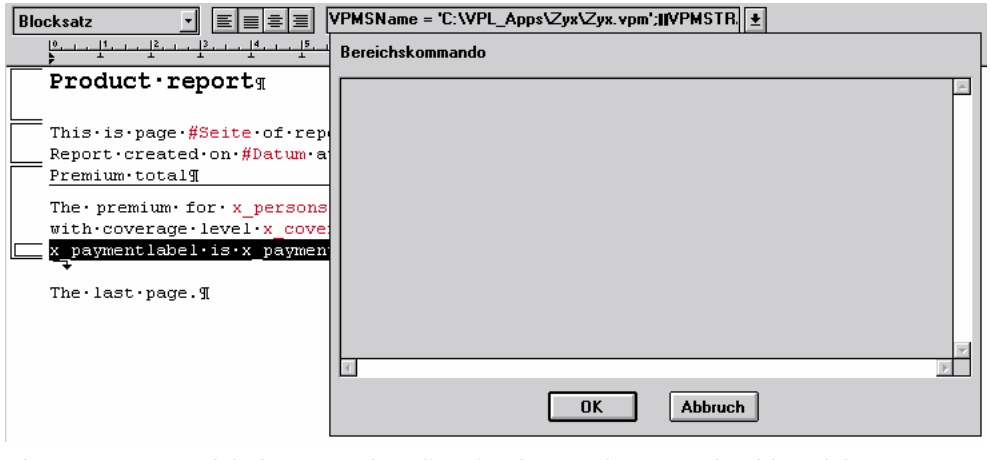


Figure 42.3. Bereichskommando editor for the newly created subbereich

42.11. Enter the following text in the editor:

```
if COMPUTE("a_Payment.visible");
x_payment = EF_Payment;
x_paymentlabel = COMPUTE("a_Payment.label");
```

42.12. Click **OK**.

42.3. Specify datafields (RE)

42.13. Specify **x_paymentlabel** as a datafield.

42.14. Specify **x_payment** as a datafield.

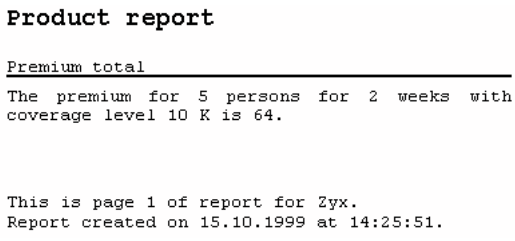
42.15. Save the report (Zyx35.cat).

42.4. Test (VF,RV)

42.16. Open the **Zyx consultation**.

42.17. Select **10 K**.

42.18. Print the report. Note that the Payment label and field are not printed.


Product report

Premium total

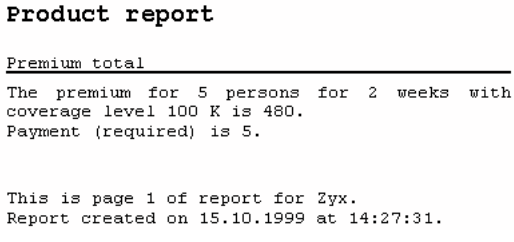
The premium for 5 persons for 2 weeks with coverage level 10 K is 64.

This is page 1 of report for Zyx.
Report created on 15.10.1999 at 14:25:51.

Figure 42.4. Payment label and field are not printed

42.19. Select **100 K**.

42.20. Print the report. Note that the Payment label and field are printed.


Product report

Premium total

The premium for 5 persons for 2 weeks with coverage level 100 K is 480.
Payment (required) is 5.

This is page 1 of report for Zyx.
Report created on 15.10.1999 at 14:27:31.

Figure 42.5. Payment label and field are printed

43. Indexes and recursion (W,D)

Indexed properties can be used to for calculations such as annual interest. The index used must be unique; however, it doesnt have to be declared.

43.1. Add (starting) capital, years, and (interest) rate attributes (W)

43.1. Create the following attributes:

43.1.1. **a_Capital**

43.1.2. **a_Years**

43.1.3. **a_Rate**

43.2. Add (for each year) indexed (accumulated) capital and annual interest properties (W)

43.2. Create property **p_Interest(year)** with the following definition:

```
p_Capital(year-1) * a_Rate / 100
```

43.3. Create property **p_Capital(year)** with the following definition:

```
if (year=0;
   a_Capital;
   p_Capital(year-1) + p_Interest(year))
```

43.3. Add property for total (end capital) (W)

43.4. Create property **p_Total** with the following definition:

```
round(p_Capital(a_Years);2)
```

43.5. Save the model (Zyx36.pms).

43.6. Create a runtime model.

43.4. Test (W,WT)

43.7. Create a new test.

43.8. Enter **p_Total** in the **Compute** field.

43.9. Click **Compute**. The **a_Years** entry field appears.

43.10. Enter **2**.

43.11. Click **Compute**. The **a_Capital** entry field appears.

43.12. Enter **100**.

43.13. Click **Compute**. The **a_Rate** entry field appears.

43.14. Enter **10**.

43.15. Click **Compute**. The result of **121** appears.

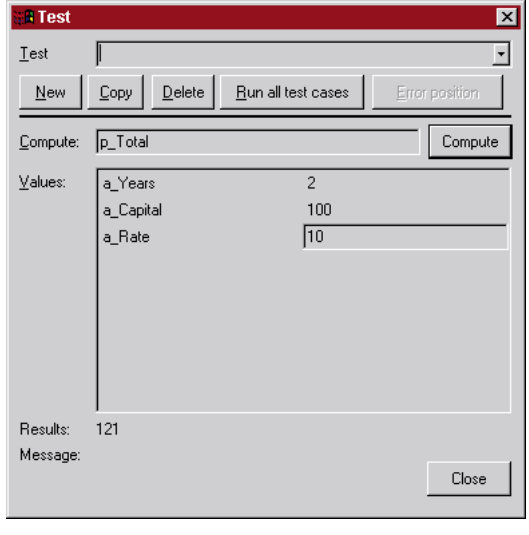


Figure 43.1. Interest calculation in the Workbench test

43.16. Enter **test3** as the name of the test.

43.17. Close the test.

43.5. Add attributes, property to layout (D)

43.18. Rename workarea **Other information** to **Capital**.

43.19. Rename the first page to **Entry fields**.

43.20. Add labels and entry/result fields for the following attributes/properties:

43.20.1. **a_Capital**

43.20.2. **a_Years**

43.20.3. **a_Rate**

43.20.4. **p_Total**

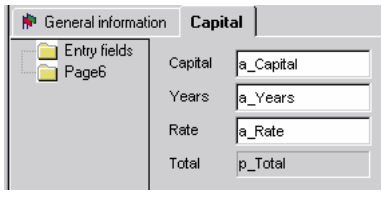


Figure 43.2. Attributes and properties for for capital in layout

43.21. Save the layout (Zyx36.vpl).

43.6. Test (D,DT)

43.22. Test the layout.

43.23. For **Capital**: Enter **100**.

43.24. For **Years**: Enter **2**.

43.25. For **Rate**: Enter **10**.

43.26. Click to change the focus.

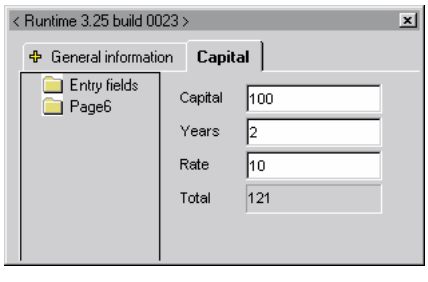


Figure 43.3. Test layout for capital

43.27. Close the test.

44. Grid (W,D)

The information in the previous chapter can be displayed in a grid.

44.1. Add attribute for starting year (W)

44.1. Rename the second page to **Grid**.

44.2. Add attribute **a_YearStart** with property **default** with value **1**.

44.2. Add property for computing next year (key for the grid) (W)

44.3. Add **Product1** property **p_YearNext(year)** with following definition:

```
year + 1
```

44.4. Save the model (Zyx37.pms).

44.5. Create a runtime model.

44.3. Add grid to the layout (D)

44.6. Rename the second page to **Grid**.

44.7. Add a **Grid list** () to the layout.

44.8. Name the list **GL_Capital**.

44.9. Set property **Column** to **4**.

44.10. Set **Iteration Type** to **Attribute**.

44.11. Set **Attribute** to **a_Years**.

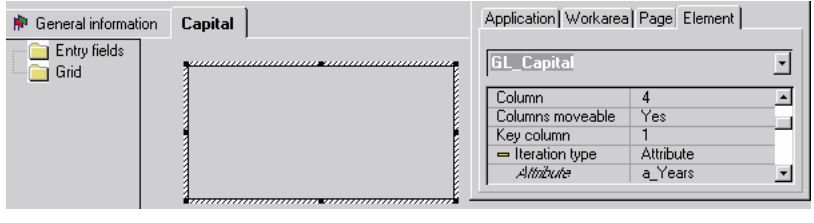


Figure 44.1. Layout with grid list

44.4. Column 1 settings (D)

44.12. Select from **Property for**: **Column 1**.

44.13. Set **Title** to **Year**.

44.14. Set **Width** to **100**.

44.15. Set **Usage** to **Input**.

44.16. Set **Attribute** to **a_YearStart**.

44.17. Set **Next line** to **2**.

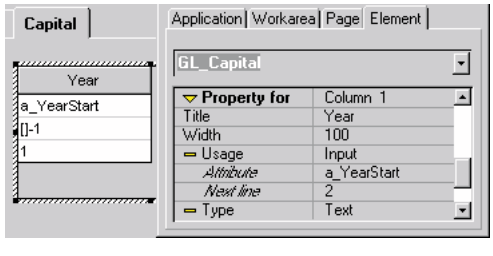


Figure 44.2. Column 1 settings

44.5. Column 2 settings (D)

44.18. Set **Property for** to **Column 2**.

44.19. Set **Title** to **(internal)**.

44.20. Set **Width** to **100**.

44.21. Set **Usage** to **Output**.

44.22. Set **Property** to **p_YearNext()**.

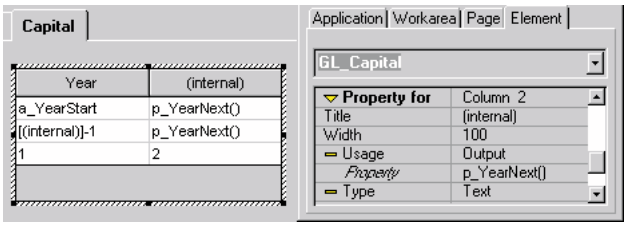


Figure 44.3. Column 2 settings

44.6. Column 3 settings (D)

44.23. Set **Property for** to **Column 3**.

44.24. Set **Title** to **Interest**.

44.25. Set **Width** to **100**.

44.26. Set **Usage** to **Output**.

44.27. Set **Property** to **p_Interest()**.

44.28. Set **Type** to **Currency**.

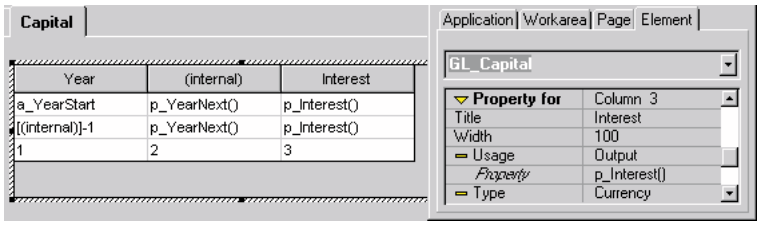


Figure 44.4. Column 3 settings

44.7. Column 4 settings (D)

44.29. Set **Property for** to **Column 4**.

44.30. Set **Title** to **Capital**.

44.31. Set **Width** to **100**.

44.32. Set **Usage** to **Output**.

44.33. Set **Property** to **p_Capital()**.

44.34. Set **Type** to **Currency**.

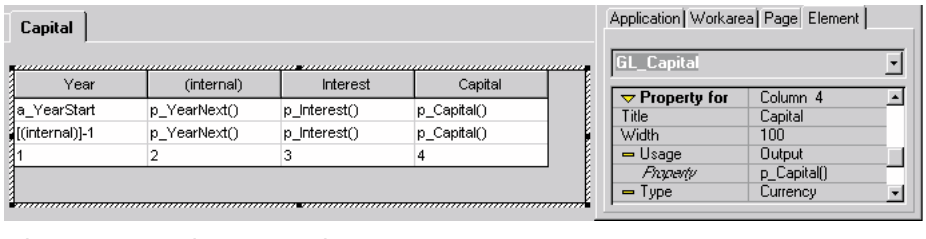


Figure 44.5. Column 4 settings

44.35. Save the layout (Zyx37.vpl).

44.8. Test (DT)

44.36. Test the layout.

44.37. In workarea **Capital** page **Entry fields**: Enter the following information:

44.37.1. For **a_Capital**: **100**

44.37.2. For **a_Years**: **10**

44.37.3. For **a_Rate**: **10**

44.38. Select the page **Grid**.

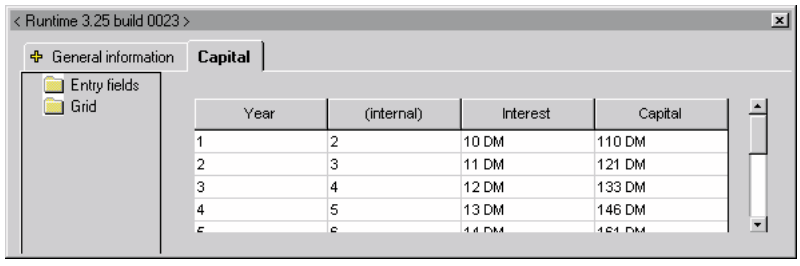


Figure 44.6. Results in the grid

44.39. Close the test dialog.

45. Graphics element (D)

Elements that are displayed in a grid can also be displayed in a graphics element.

45.1. Add page with graphics element to the layout (D)

45.1. Add a page to workarea **Capital** with title **Graphic**.

45.2. Add a **Graphics elementt** () to the layout.

45.3. Name the list **GE_Capital**.

45.4. For **Indentification Grid**: Select **GL_Capital**.

45.5. For **X-axe name**: Enter **Year**.

45.6. For **Y-axe name**: Enter **DM**.

45.7. For **Series**: Enter **2**.

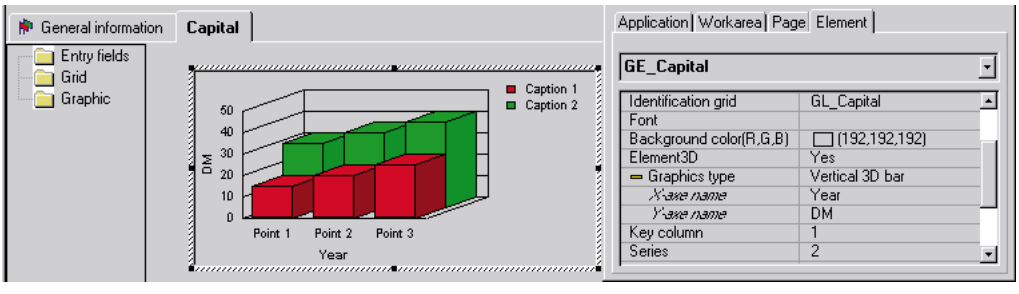


Figure 45.1. Graphics element settings

45.2. Set properties for Series 1 (Interest) (D)

45.8. Set **Data column** to **3**.

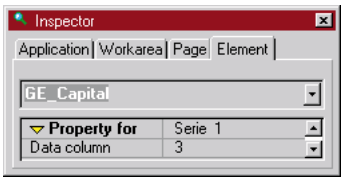


Figure 45.2. Settings for Series 1

45.3. Set properties for Series 2 (Capital) (D)

45.9. Set **Data column** to **4**.

45.10. Save the layout (Zyx38.vpl).

45.4. Test (D,DT)

45.11. Test the layout.

45.12. In workarea **Capital** page **Entry fields**: Enter the following information:

45.12.1. For **a_Capital**: **100**

45.12.2. For **a_Years**: **5**

45.12.3. For **a_Rate**: **50**

45.13. Select the page **Graphic**.

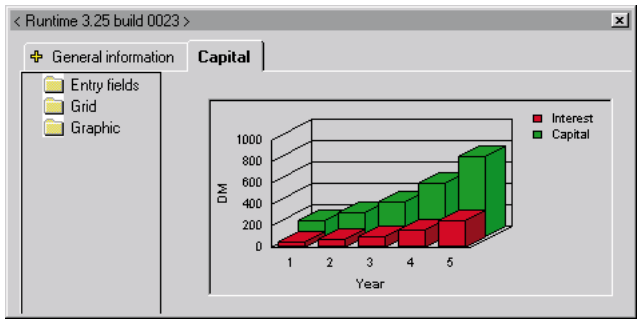


Figure 45.3. Results in the graphic

45.14. Close the test dialog.

46. Dynamic arrays in the report ??

46.1.
46.2.

47. Create multiple subproducts (W)

47.1. Create ProductMain; make Product1 a subproduct

47.1. In the **Products** window: Create product **ProductMain**.

47.2. In the toolbar: Click on the **Move branch up** icon (). **ProductMain** is moved to the top of the **Products** window.

47.3. Select **Product1**.

47.4. Right-click.

47.5. Select **Demote**. Note that Product1 is now a subproduct of ProductMain.

Figure 47.1. Product1 as a subproduct of ProductMain

47.2. Create ProductMain subproduct Product2; add property p_Premium

47.6. In the **Products** window: Create product **Product2** as a subproduct of **ProductMain**.

47.7. Add property **p_Premium** with value **1000**.

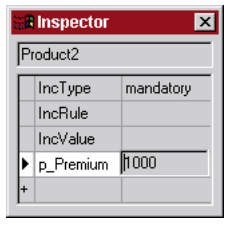


Figure 47.2. p_Premium definition for Product2

47.8. Save the model (Zyx40.pms).

47.9. Generate a runtime model.

47.3. Test (W)

47.10. Click on the **Test product model** icon. The previous test appears.

47.11. Click **Compute**. Note that the value for p_Premium is the sum of the values for p_Premium in each subproduct.

47.12. Close the test dialog.

48. Create multiple subproducts in layout (D)

48.1. Change tab names

48.1. In the **Designer**: Select page **Area1**.
48.2. In the **Inspector**: Select tab **Workarea**.
48.3. Change **Title** to **Personal info**.
48.4. Click **Enter**.
48.5. Select page **Area2**.
48.6. In the **Inspector**: Select tab **Workarea**.
48.7. Change **Title** to **Product info**.
48.8. Click **Enter**.

48.2. Change Product info workarea style to frame

48.9. Select the **Product info** workarea.
48.10. Change property **Style** to **Frame**. Note that a default page is also created.

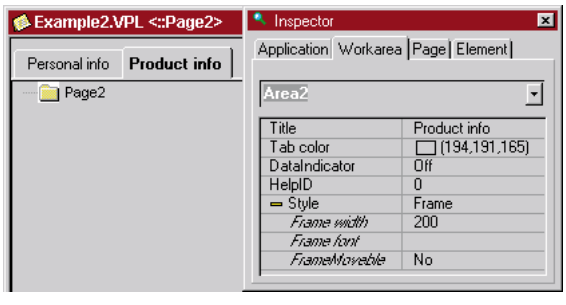


Figure 48.1. Workarea Product info with style frame

48.3. Rename page; create Product1, Product2 pages

48.11. Select tab **Page**.
48.12. Change **Title** to **All products**.

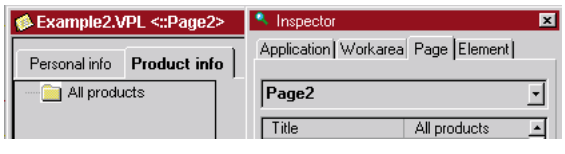


Figure 48.2. Changing name of page

48.13. In the layout (not in the inspector): Right-click on **All products**. A pop-up window appears.
48.14. Select **Insert subnode**. A subnode for node All products appears.
48.15. Change the **Title** of the subnode to **Product1**.
48.16. Insert another subnode for **All products**.
48.17. Change the **Title** of the subnode to **Product2**.

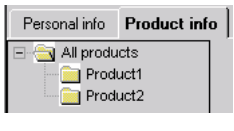


Figure 48.3. Product1 and Product2 subnodes of All products

48.4. Move / Copy required components from Workarea Personal info to required workareas

48.18. Select workarea **Personal info**.
48.19. Click and hold the **CTRL-key**.
48.20. Right-click on label **Premium**.

48.21. Right-click on result field **p_Premium**..

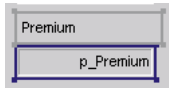


Figure 48.4. Selected label and result field in the workarea Personal info

48.22. Press **CRTL-X**. Note that the selected elements disappear.

48.23. Select page **All products**.

48.24. Click **CTRL-V**. Note that the components are copied to the same location in the new page as they were in the previous page.

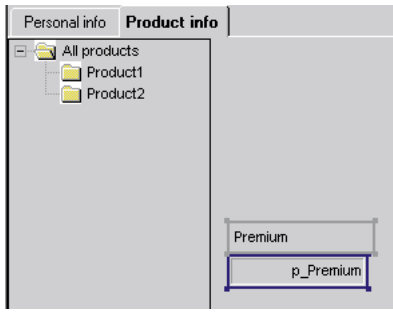


Figure 48.5. Copied elements in page All products

48.25. Copy the same elements to page **Product1**.

48.26. Copy the same elements to page **Product2**.

48.27. Move the following elements from workarea **Personal info** to page **Product1**.

- Label **Weeks of coverage**
- Combo box for **a_Weeks**
- Both **radiogroups**

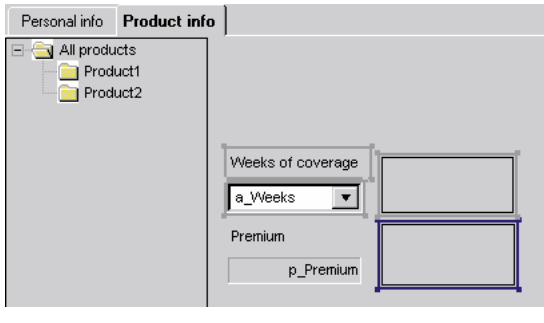


Figure 48.6. Moved elements from workarea Personal info to page Product1

48.28. Rearrange the components on all pages.

48.5. Test (D)

48.29. Click **Test application** (save the layout (Zyx41.vpl)). The following test dialog appears:

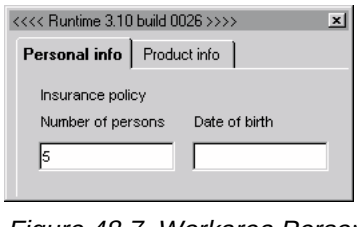


Figure 48.7. Workarea Personal info in the Designer test

48.30. Enter **1.1.1900** for **Date of birth**.

48.31. Select workarea **Product info**. Note that page **Product1** is automatically selected (required data must be entered in this page).

48.32. Select **10K**. Note that the result of 1040 for p_Premium is displayed..

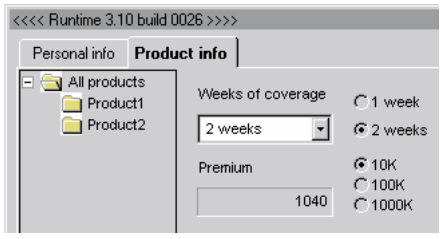


Figure 48.8. p_Premium result in workarea Product info page Product1

48.33. Select **Product2**. Note the p_Premium result.

48.34. Select **All products**. Note the p_Premium result.

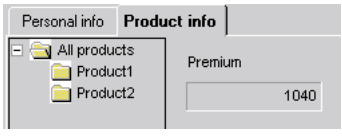


Figure 48.9. p_Premium result in the page All products

48.35. Close the test dialog.

48.6. Test (VF)

48.36. Test as in the previous chapter. Note that the functionality has not changed.

48.7. View a report (VF, RV)

48.37. Test as in the previous chapter. Note that the functionality has not changed.

49. Add multiple subproducts to report (RE)

49.1. Make Bereichskommando text kommando for entire document

49.1.1. Copy bereichskommando text

49.1. In the **Report Editor**: Click within the Bereich.

49.2. Open the **Bereichskommando** dialog for the Bereich. The text in the dialog is already selected.

49.3. Press **CTRL-C**.

49.4. Close the **Bereichskommando** dialog.

49.5. Place the cursor anywhere in the text outside the Bereich.

49.6. Open the **Bereichskommando** dialog (it is empty).

49.7. Press **CTRL-V**. The Bereichskommando text is copied.

49.8. Press **OK**.

49.1.2. Delete previous bereich

49.9. Place the cursor anywhere in the original bereich.

49.10. Select **Bereich / Löschen**.

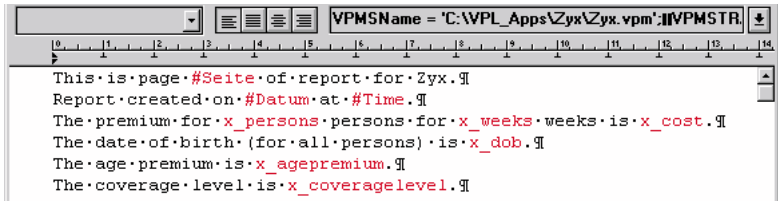


Figure 49.1. Bereichskommando for entire document

49.2. Change text

49.11. Replace the text in the dialog with the following text. NOTE: If cut and paste doesnt work, try restarting the RE.

```
Product report
Page #Seite. Created on #Datum at #Time.
Personal Info
x_persons persons.
Date of birth (for all persons) is x_dob.
Age premium is x_agepremium.
All products
Premium is x_cost.
Product1
Number of weeks is x_weeks.
Coverage level is x_coveragelevel.
Product2
(no input required for Product2).
```

49.12. Select the entire text.

49.13. From the **Character format** Combo box: Select **Mono**.

49.14. From the **Paragraph format** Combo box: Select **Blocksatz**.

Figure 49.2. Character and paragraph specification the RE

49.3. Specify header text

49.15. Select the following line:

```
Product report.
```

49.16. Select **Bereiche / Kopfzeilen**. Note that the text is now marked as a bereich.

49.4. Specify footer text

49.17. Select the following line:

```
Page #Seite. Created on #Datum at #Time.
```

49.18. Select **Bereiche / Fusszeilen**. Note that the text is now marked as a bereich.

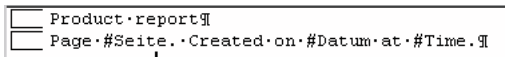


Figure 49.3. Header and footer text

49.5. Create new character style

NOTE: RE does not allow changing the standard character format for a paragraph format. Thus, in order to change the format of characters in a paragraph, the character format must be changed (as demonstrated in this section).

49.19. Select the following line (the header text):

```
Product report.
```

49.20. Select **Format / Zeichen**. The **FormatAuswahl** dialog appears:

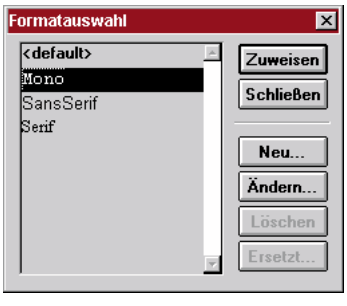


Figure 49.4. FormatAuswahl dialog

49.21. Click **Neu...**. The standard **Schriftart** dialog appears:

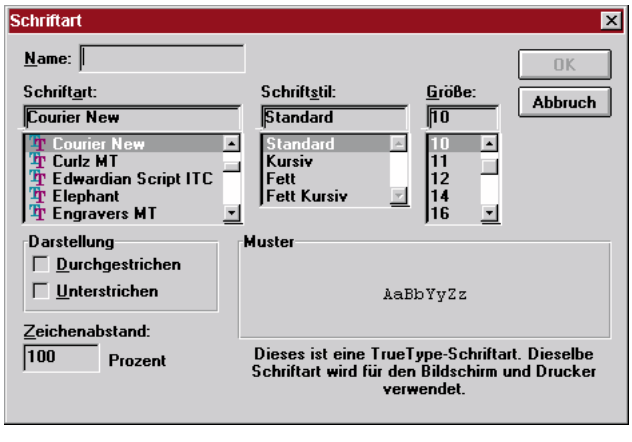


Figure 49.5. Standard Schriftart dialog

49.22. Select **Schriftstil Fett**.

49.23. Select **Grosse 14**.

49.24. In the **Name:** entry field: Enter **Header**.

49.25. Click **OK**. The **Formatauswahl** dialog regains the focus.

49.26. Click **Zuweisen**. Note the new text style.

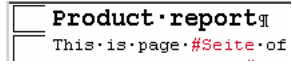


Figure 49.6. New Header style text

49.6. Create new paragraph style

49.27. Place the cursor within the following text:

```
Personal Info
```

49.28. Select **Format / Absatz**. The **Formatauswahl** dialog appears.

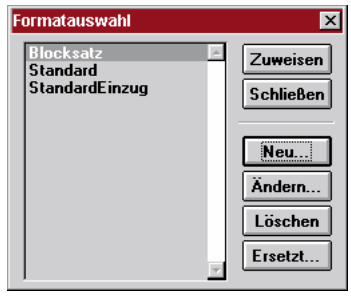


Figure 49.7. Formatauswahl dialog

49.29. Click on **Neu...**. The **Absatzformat** dialog appears:

Figure 49.8. Absatzformat dialog

49.30. Select **Rahmen**. The **Absatzrahmen** dialog appears.

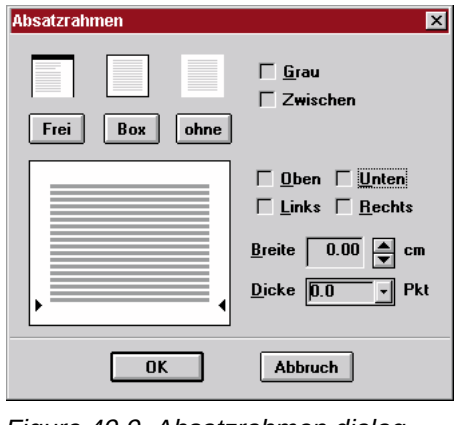


Figure 49.9. Absatzrahmen dialog

49.31. Click the checkbox **Unten**.

49.32. From the **Dicke** combo box: Select **1**.

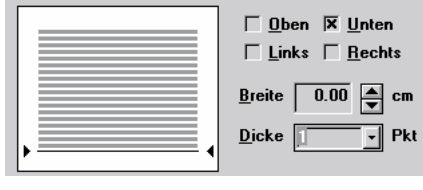


Figure 49.10. Selecting the box lines and thickness

49.33. Select **OK**. The **Absatzformat** dialog regains the focus.

49.34. Change **Zeilenabstände / Unten** to **0.20 cm**.
49.35. In the **Name** entry field: Enter **ZyxHeading1**.
49.36. Select **OK**. The **Formatauswahl** dialog regains the focus.
49.37. Select **Zuweisen**. Note the new format.

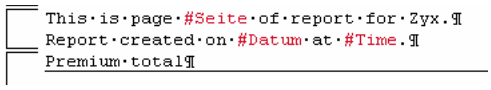


Figure 49.11. ZyxHeading1 format

49.38. Click on the **Paragraph formats** combo box. Note that the new format is included.

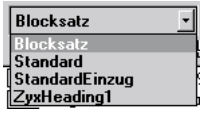


Figure 49.12. ZyxHeading1 in the paragraph formats combo box

49.7. Insert page break

49.39. Place the cursor at the beginning of the following line:

```
All products
```

49.40. Press **CTRL-RET**. Note that a page break has been inserted.

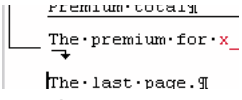


Figure 49.13. Page break

49.8. Insert page breaks and change paragraph formats

49.41. Insert page breaks before the following lines:

```
Product1
Product2
```

49.42. Change the format for the above lines to **ZyxHeading1**.Specify datafields
49.43. Specify the following text as datafields:

```
#Seite
#Datum
#Time
x_persons
x_dob
x_agepremium
x_cost.
x_weeks.
x_coveragelevel.
```

Figure 49.14. Text with line breaks, paragraph formats, and datafields

49.44. Save the report.

49.9. Test (VF)

49.45. Restart **VFrame**.

49.46. Open the **Zyx consultation**.

49.47. Enter the required information (date of birth, coverage level).

49.48. Print the report. Page 1 of the report appears:

Figure 49.15. Page 1 of the report

Note that the footer at the bottom of the page.

49.49. Close the report. Do NOT close VFrame.

49.10. Modify page format (RE)

The page format will be modified to make it easier to read the pages on the screen.

NOTE: Before printing to a printer, the format should be changed back to A4 (by clicking button **A4** in the Dokument - Einstellungen dialog).

49.50. In the **Report Editor**: Select **Format / Dokument**. The **Dokument - Einstellungen** dialog appears:

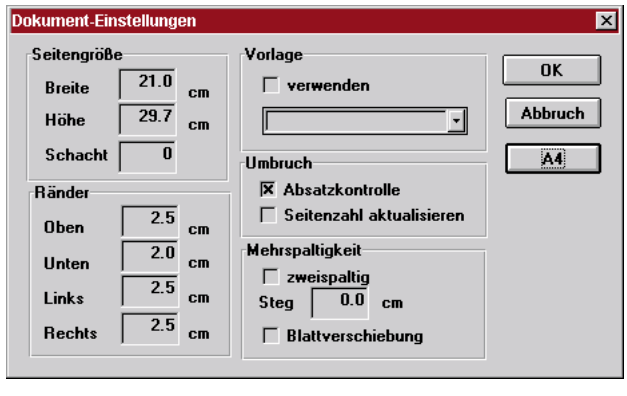


Figure 49.16. Dokument - Einstellungen dialog

49.51. Change the following:

49.51.1. **Breite**: **12**.

49.51.2. **Höhe**: **5**.

49.51.3. **Oben**: **0.5**.

49.51.4. **Unten**: **0.5**.

49.51.5. **Links**: **0.5**.

49.51.6. **Rechts**: **0.5**.

49.52. Click **OK**.

49.53. Save the report (Zyx42.cat).

49.11. Test (VFrame)

NOTE: VFrame does not need to be restarted after changes have been made to the report.

49.54. Print the report. Page 1 of the report appears:

Figure 49.17. Page 1 of the report with new page format

49.55. Click on the **next page** icon () to display pages 2, 3, and 4:

Figure 49.18. Page 2 of the report with new page format

Figure 49.19. Page 3 of the report with new page format

Figure 49.20. Page 4 of the report with new page format

49.56. Click the **First page** () icon to return to the first page.

49.57. Close the report.

50. Create functions (W)

50.1. Add attribute a_DOB

50.1. Add attribute **a_DOB**.

50.2. Add function f_Age(a_DOB) with functions years, today

50.2. In the window **Functions**: Right-click. The **New** pop-up window appears.
50.3. Select **New**.
50.4. Type in the following text: **f_Age(a_DOB)**.
50.5. Double-click on **f_Age(a_DOB)** to open the **Editor**.

Figure 50.1. Editor window

50.6. In the lower window of the Editor: Enter the following code:

```
years(today("d.m.y") ) - years(a_DOB).
```

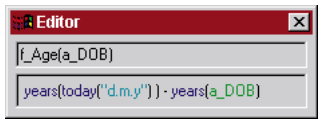


Figure 50.2. Editor window with code

50.3. Create table t_AgePremium

50.7. Create table **t_AgePremium** with the following Keys / Values:
50.7.1. 0 / 1
50.7.2. **50 / 2**

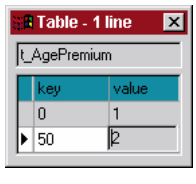


Figure 50.3. t_AgePremium

50.4. Add function f_AgePremium with function lookup

50.8. Create function **f_AgePremium** with the following code:

```
lookup( t_AgePremium, f_Age(a_DOB) )
```

50.5. Edit calculation of p_Premium

50.9. For Product **Product1**: Change the code for property **p_Premium** to the following:

```
a_Persons * a_Weeks * f_AgePremium
```

50.10. Save the model (Zyx43.pms).
50.11. Create a runtime model.

50.6. Test (W)

50.12. Click on the **Test product model** icon.
50.13. Select **test1**. Note the entries for a_Persons and a_Weeks.
50.14. Click **Compute**. An entry field for **a_DOB** appears.
50.15. Enter **1.1.1900**.

50.16. Click **Compute**. Note the result of **12**:

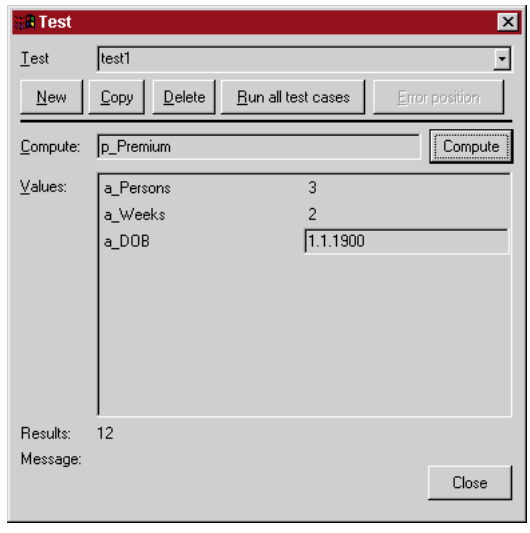


Figure 50.4. Test window with date of birth premium

50.17. Close the Test dialog.

50.7. Add label / edit field in layout (D)

50.18. In the **Designer**: Add a **label**.

50.19. Set the label **Title** to **Date of birth**.

50.20. Add an **edit field**.

50.21. In the Inspector: Open the Combo box for **Source**. Note that **a_DOB** is not present in the list. The layout must reopened after an attribute has been added to the model.

50.22. Close (and save) **Zyx.VPL**.

50.23. Open **Zyx.VPL**.

50.24. Select the **entry field**.

50.25. Set the **Source** as **a_DOB**.

50.26. Set **Compute** to **OnChange**.

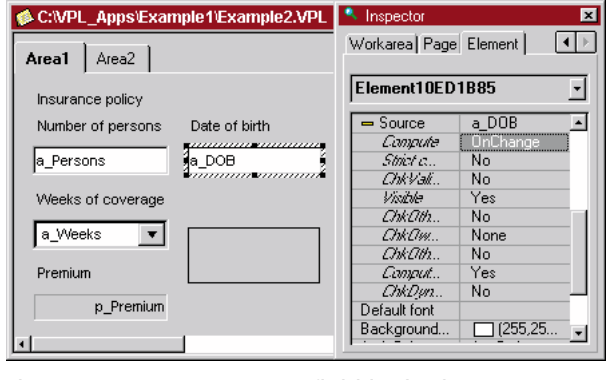


Figure 50.5. a_DOB entry field in the layout

50.8. Test (Designer)

50.27. Click on the **Test Application** button in the toolbar.

50.28. Click **Yes** to save the application (Zyx43.vpl). The following dialog appears.

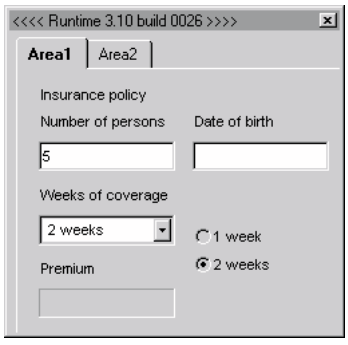


Figure 50.6. a_DOB entry field in the layout test dialog

50.29. Test the layout. Note that the Date of Birth input has no effect on the results. This is because Designer must be restarted after the model has been changed.

50.30. Restart **Designer**.

50.31. Open **Zyx.VPL**.

50.32. Click the **Test application** icon. The test dialog appears.

50.33. Enter **1** in the **Date of birth** entry field. Note that no result or error message appear in the test dialog; however, the message **Date expected instead of '1'** appears in the Designer status bar.

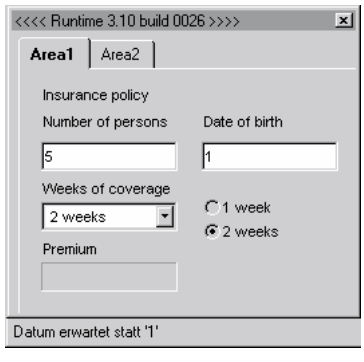


Figure 50.7. Message in the Designer status bar about Test dialog input

50.34. Add **.1.1900** to the **Date of birth** entry field. The result of 20 is displayed:

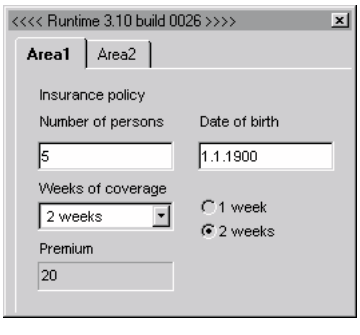


Figure 50.8. Proper result in test dialog after valid date of birth entered

50.35. Enter **1.1.1999** in the **Date of birth** entry field. Note that the Premium is now 10.

50.36. Close the test dialog.

50.9. Test (VF)

50.37. Restart **VFrame**.

50.38. Test in VFrame. Note that the results are the same as for the test in the Designer.

51. Add mutually exclusive products in model (W)

51.1. Add attribute a_ProductType; table t_ProductType

51.1. Add attribute **a_ProductType**.

51.2. For the attribute: Add property **table** with value **"t_PropertyType"** (include the quotation marks).

51.3. Add table **t_PropertyType**.

51.4. For table **t_PropertyType**: Add the following keys / values:

51.4.1. **1** / **Product1**.

51.4.2. **2** / **Product2**.

51.2. Add inclusion rules for Product1

51.5. Open the inspector for **Product1** (double-click on **Product1**).

51.6. For property **IncType**: Select **mutually exclusive**.

51.7. For property **IncRule**: Enter **a_ProductType**.

51.8. For property **IncValue**: Enter **1**.

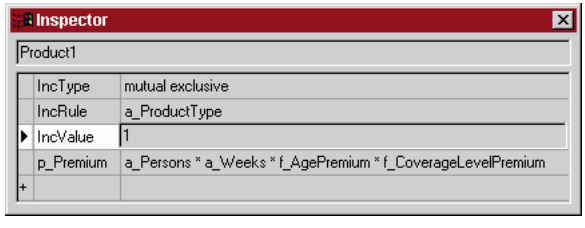


Figure 51.1. Inclusion rules for Product1

51.3. Add inclusion rules for Product2

51.9. Open the inspector for **Product2**.

51.10. For property **IncType**: Select **mutually exclusive**.

51.11. For property **IncRule**: Enter **a_ProductType**.

51.12. For property **IncValue**: Enter **2**.

51.13. Save the model (Zyx44.pms).

51.4. Test (W)

51.14. Click on the **Test product model** icon. The previous test appears.

51.15. Click **Compute**. A drop-list box for **a_ProductType** appears:

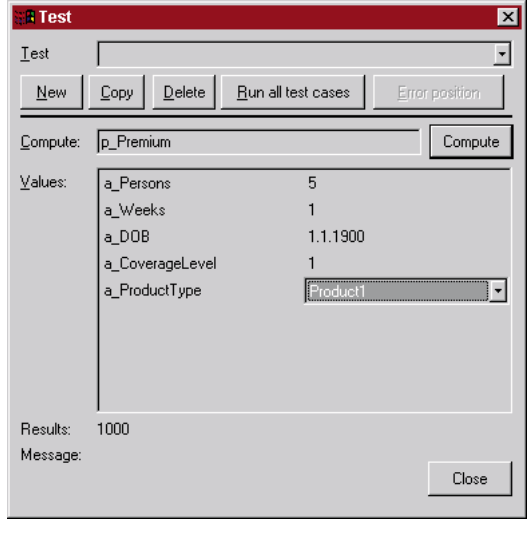


Figure 51.2. Drop-list box for a_ProductType in test window

51.16. Select **Product1**.

51.17. Click **Compute**. Note that the result includes only Product1.

51.18. Select **Product2**.
51.19. Click **Compute**. Note that the result (1000) includes only Product2.
51.20. Close the test dialog.

52. Add mutually exclusive products in layout (W, D)

52.1. Create properties for radio buttons (W)

52.1. In the **Workbench**: Create attribute **a_Product1**.

52.2. Add to **a_Product1** property **default** with the following value:

```
if (a_ProductType=1;1;0)
```

52.3. Create attribute **a_Product2**.

52.4. Add to **a_Product2** property **default** with the following value:

```
if (a_ProductType=2;1;0)
```

52.2. Create property dynamic for a_ProductType (W)

52.5. Add to **a_ProductType** property **dynamic** with the following value:

```
"a_Product1;a_Product2"
```

52.6. Save the model (Zyx45.pms).

52.7. Create runtime model.

52.3. Add radio group for selection of product (D)

52.8. Restart **Designer**.

52.9. In **Designer**: For workarea **Product info** page **All products**: Add a **radio group**.

52.10. Specify the **Source** of the **radio group** as **a_ProductType**.

52.11. Specify **ChkDynamic** as **Yes**.

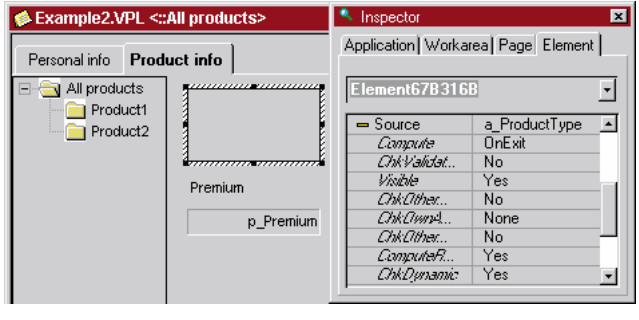


Figure 52.1. Radio group for product selection in workarea Product info page All products

52.4. Add radio button for visualizing page Product1 / devisualizing page Product2

52.12. Add a radio button ()to workarea **Product info** page **All products**.

52.13. Specify **Title** as **a_P1**.

52.14. Specify **Source** as **a_Product1**.

52.15. Specify **Visible** as **No**.

52.16. Enter for **Group** text **group1**.

52.17. Click on the text **<not selected>** to the right of text **Visualize**. A directory search button appears.

52.18. Click on the directory search button. The **Visualize** dialog appears:.

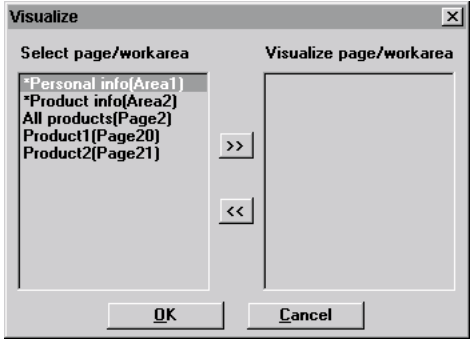


Figure 52.2. Visualize dialog for radio button a_Product1

52.19. Select **Product1**.
52.20. Click **>>**.
52.21. Click **OK**. Note that **Visualize** is now specified as "<selected>".
52.22. Click on **Devisualize**. A directory search button appears.
52.23. Click on the directory search button. The **Devisualize** dialog appears.

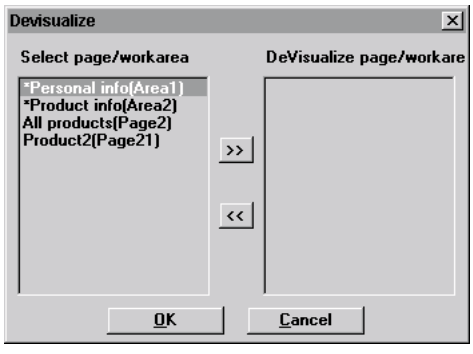


Figure 52.3. Devisualize dialog for radio button a_Product1

52.24. Select **Product2**.
52.25. Click **>>**.
52.26. Click **OK**.

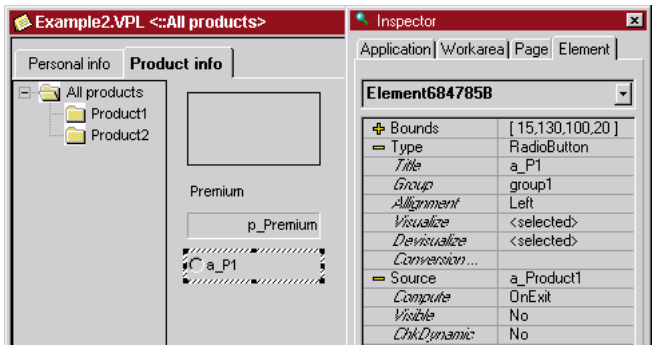


Figure 52.4. Settings for radio button a_P1

52.5. Add radio button for visualizing page Product2 / devisualizing page Product1

52.27. Add a radio button to workarea **Product info** page **All products**.
52.28. Specify **Title** as **a_P2**.
52.29. Specify **Source** as **a_Product2**.
52.30. Specify **Visible** as **No**.
52.31. For **Group**: Select **group1** from the Combo box.
52.32. Click on **Visualize**. A directory search button appears.
52.33. Click on the directory search button. The **Visualize** dialog appears.
52.34. Select **Product2**.
52.35. Click **>>**.
52.36. Click **OK**.
52.37. Click on **Devisualize**. A directory search button appears.
52.38. Click on the directory search button. The **Devisualize** dialog appears.
52.39. Select **Product2**.
52.40. Click **>>**.

52.41. Click **OK**.

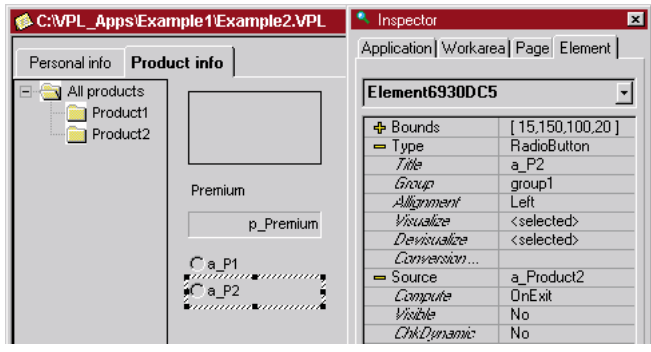


Figure 52.5. Settings for radio button a_P2

52.6. Test (D)

52.42. Click on **Test Application** (save the version (Zyx45.vpl)). The workarea **Personal info** appears.

52.43. Enter **Date of birth** as **1.1.1900**.

52.44. Select workarea **Product info**.

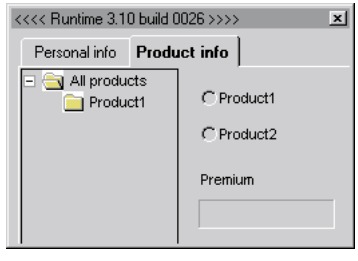


Figure 52.6. Workarea Product info page All products with radio button group for selecting product

52.45. Select radio button **Product1**. The page **Product1** is displayed ("visualized").

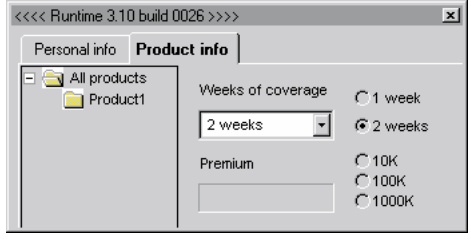


Figure 52.7. Page Product1 displayed

52.46. Select **10K**. Note the **p_Premium** is **40**.

52.47. Select page **All products**.

52.48. Select radio button **Product2**. Note that **p_Premium** is **1000** and the visible subpage is **Product2**.

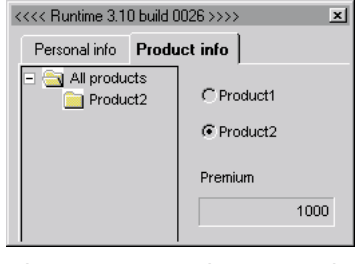


Figure 52.8. Workarea Product info after Product2 selected

52.49. Close the test dialog.

52.7. Test (VF)

52.50. Test in VFrame (do not print). Note that the results are the same as for the test in Designer.

53. Add mutually exclusive products in Report (RE)

53.1. Create bereich for Product1 page

53.1. In the **RE**: Select the **New line** character before the page for **Product1** and the text for the **Product1** page.

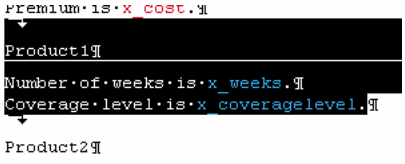


Figure 53.1. Select page break and Product1 page text in RE

53.2. Select **Bereich / New**. The Bereichskommando dialog appears.

53.3. Enter the following text:

```
if (COMPUTE("a_ProductType") = 1);
```

53.4. Click **OK**.

53.2. Create bereich for Product2 page

53.5. Select the **New line** character before the page for **Product2** and the text for the **Product2** page.

53.6. Select **Bereich / New**. The Bereichskommando dialog appears.

53.7. Enter the following text:

```
if (COMPUTE("a_ProductType") = 2);
```

53.8. Click **OK**.

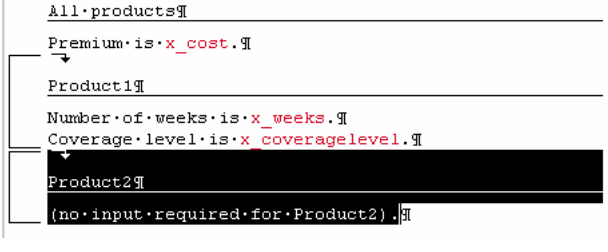


Figure 53.2. 2 new bereichs in the report

53.9. Save the report (Zyx46.cat).

53.3. Test (VF)

53.10. In **VFrame**: Select **Product1**.

53.11. Print the report. Note that the report does not contain a page for Product2.

53.12. Close the report.

53.13. In **VFrame**: Select **Product2**.

53.14. Print the report. Note that the report does not contain a page for Product1.

53.15. Close the report.

54. Add optionally inclusive products to model (W)

54.1. Specify optional inclusion for Product2

54.1. In **Workbench**: Open the inspector for **Product2**.

54.2. Set **IncType** to **Optional**.

54.3. Set **IncRule** to the following code:

```
f_Age(a_DOB) > 100
```

54.4. Set **IncValue** to blank.

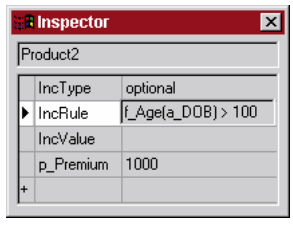


Figure 54.1. Optional inclusion settings for Product2

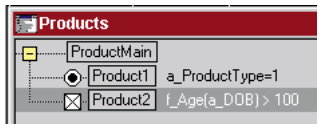


Figure 54.2. Product2 in the Products window

54.2. Specify dynamic for a_DOB

54.5. For **a_DOB**: Create property **dynamic** with value **"a_Product2"** (include quotation marks).

54.3. Specify default for a_Product2

54.6. For **a_Product2**: Modify property **default** to the following:

```
if (f_Age(a_DOB)>100;1;0)
```

54.7. Save the model (Zyx47.pms).

54.8. Create a runtime model.

54.4. Test (W)

54.4.1. Product1 only

54.9. Create a new test for computing **p_Premium** with the following:

54.9.1. **a_ProductType** = **Product1**

54.9.2. **a_Persons** = **5**

54.9.3. **a_Weeks** = **1 week**

54.9.4. **a_DOB** = **1.1.1900**

54.9.5. **a_CoverageLevel** = **10K**

54.10. Click **Compute**. Note that the result of **10** includes only Product1 (age is < 100).

54.4.2. Product1 and Product2 (age>100)

54.11. Set **a_DOB** to **1.1.1800**.

54.12. Click **Compute**. Note the result **1010** includes Product1 andProduct2.

54.4.3. Product2 only (age <100)

54.13. Set **a_ProductType** to **Product2**.

54.14. Set **a_DOB** to **1.1.1900**.

54.15. Click **Compute**. Note the error **"-> Attribute / property p_Premium not defined"**.

54.4.4. Product2 only (with age >100)

54.16. Set **a_DOB** to **1.1.1800**.

54.17. Click **Compute**. Note the result **1000** includes only Product2.

54.18. Close the test dialog.

55. Add optionally inclusive products in layout (D)

55.1. Set ChkDynamic for Date of Birth entry field to Yes

55.1. In **Designer**: In workarea **Personal info**. For entry field **Date of birth**: Set **ChkDynamic** to **Yes**.

55.2. Delete Devisualize for a_P1

55.2. In workarea **Product info** page **All products**: Delete the **Devisualize** entry for **a_P1**.

55.3. Change radio button a_P2 to checkbox; delete Devisualize for a_P2

55.3. In workarea **Product info** page **All products**: Select radio button **a_P2**.

55.4. From the **Type** Combo box: Select **CheckBox**.

55.5. Delete the **Devisualize** entry for **a_P2**.

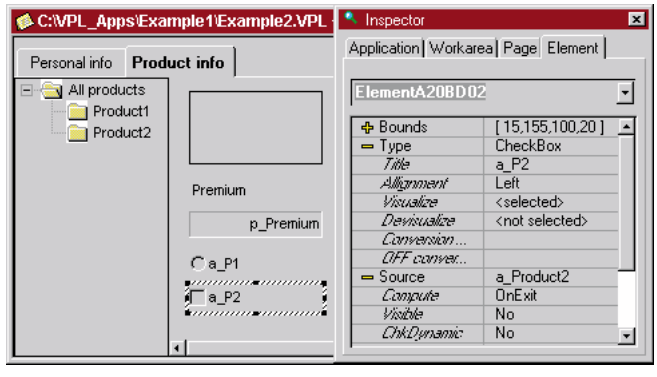


Figure 55.1. Settings for checkbox a_P2 in Designer

55.4. Test (D)

55.4.1. Date of birth < 100

55.6. Click on **Test Application** (save application (Zyx48.vpl)). The workarea **Personal info** appears.

55.7. Enter **Date of birth** as **1.1.1900** (age < 100).

55.8. Select workarea **Product info**. Note that no products are selected:

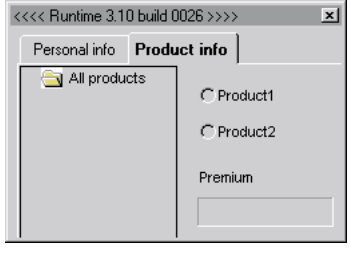


Figure 55.2. No products selected in Designer test

55.9. Select **Product2** (radio button). Note that Product2 does not appear (and no error message is generated).

55.4.2. Date of birth > 100

55.10. Enter **Date of birth** as **1.1.1800** (age > 100).

55.11. Select workarea **Product info**. Note that **Product2** is visible and **p_Premium** is displayed:

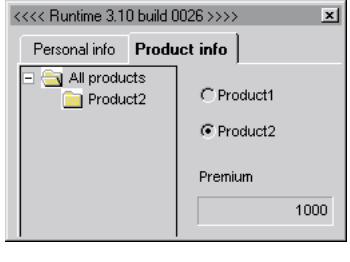


Figure 55.3. Product2 visible in Designer test

55.12. Select **Product1**. Note that both Product1 and Product2 are displayed.
55.13. Close the test dialog.

56. Add optionally inclusive products to Report (RE)

56.1. Change bereichskommando for Product2

56.1. In the **RE**: Change the bereichskommando for the Product2 bereich to the following:

```
if (COMPUTE("a_Product2") = 1);
```

56.2. Save the report (Zyx49.cat).

56.2. Test (VF)

56.3. In **VFrame**: Select **Product1**.

56.4. Enter **Date of birth** as **1.1.1900**.

56.5. Select a **Coverage Level**.

56.6. Print the report. Note that the report does not contain a page for Product2.

56.7. Close the report.

56.8. In **VFrame**: Select **Product2**.

56.9. Print the report. No valid product selection has been made; therefore, the printout is empty:

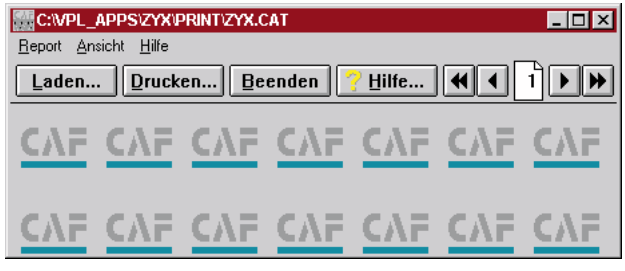


Figure 56.1. Printout for invalid product selection

56.10. Change **Date of birth** to **1.1.1800**.

56.11. Print the report. Note that the report does not contain a page for Product1.

56.12. Close the report.

57. Add multiply inclusive products to model (W)

57.1. Add Product1 property p_Premium

57.1. In the **Workbench**: For **ProductMain**: Add property **p_Premium** with the following definition:

```
p_PremiumP1 + p_PremiumP2
```

57.2. Modify Product1 for multiple inclusion

57.2. Change **IncType** for **Product1** to **multiple**.

57.3. Change **IncRule** to **a_Persons**.

57.4. Change **IncRule** to blank.

57.3. Delete / add Product1 properties

57.5. Delete property **p_Premium**.

57.6. Add property **p_PremiumP1** with the following definition:

```
a_Weeks[#a_Persons] * f_AgePremium * f_CoverageLevelPremium
```

57.7. Add property **p_PremiumEach(i)** with the following definition:

```
if(i=#a_Persons;
   p_PremiumP1;
   0)
```

57.8. Add property **DI_Product1(i)** with the following definition:

```
if(i=#a_Persons;
   if(
      not(undefined("a_Name" & "[" & #a_Persons & "]"))
      &&
      length(a_Name[#a_Persons]) > 0
      &&
      not(undefined("a_CoverageLevel" & "[" & #a_Persons & "]"))
      &&
      a_CoverageLevel[#a_Persons] >0 ;
      1;
      0);
   0)
```

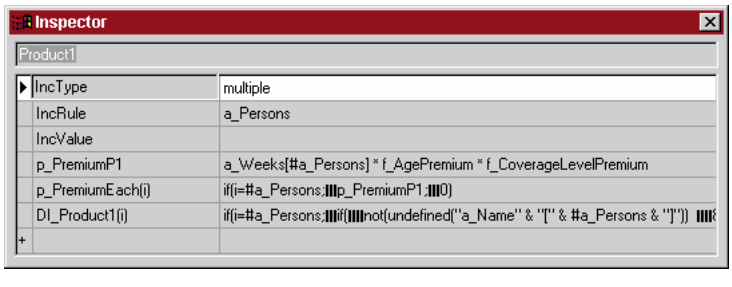


Figure 57.1. Product1 in the inspector

57.4. Modify Product2 for multiple inclusion

57.9. Change **IncType** for **Product2** to **multiple**.

57.10. Change **IncRule** to **a_Persons**.

57.11. Change **IncRule** to blank.

57.5. Delete / add Product2 properties

57.12. Delete property **p_Premium**.

57.13. Add property **p_PremiumP2** with the following definition:

```
if (f_Age(a_DOB) > 100;1000;0)
```

57.14. Add property **p_PremiumP2Each(i)** with the following definition:

```
if(i=#a_Persons;
```

```
p_PremiumP2;
0)
```

57.15. Add property **DI_Product2(i)** with the following definition:

```
if(i=#a_Persons;
  if(
    not(undefined("a_DOB" & "[" & #a_Persons & "]"))
    &&
    days(a_DOB[#a_Persons]) >0;
    1;
    0);
  0)
```

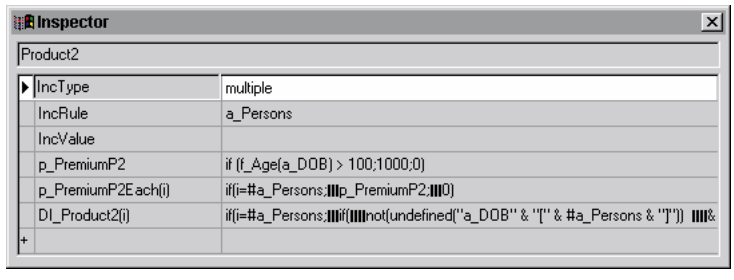


Figure 57.2. Product2 in the inspector

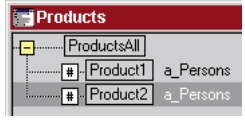


Figure 57.3. Product1, Product2 with multiple inclusion in Workbench

57.6. Add attribute a_Name

57.16. Add attribute **a_Name**.

57.7. Change defaults for a_Persons, a_CoverageLevel

57.17. For attribute **a_Persons**: Change **default** to **2**.

57.18. For attribute **a_CoverageLevel**: Add property **default** with value **1**.

57.8. Modify f_Age(a_DOB)

57.19. Change the code for **f_Age(a_DOB)** to the following:

```
years(today("d.m.y") ) - years(a_DOB[#a_Persons])
```

57.9. Modify f_CoverageLevelPremium

57.20. Change the code for **f_CoverageLevelPremium** to the following:

```
if ( a_Weeks = 1 ;
  t_CoveragePremium[a_CoverageLevel[#a_Persons]].weeks1 ;
  t_CoveragePremium[a_CoverageLevel[#a_Persons]].weeks2 )
```

57.21. Save the model (Zyx50.pms).

57.22. Create a runtime model.

57.10. Test (W)

57.23. Create a new test.

57.24. In entry field **Compute:**: Enter **p_Premium**.

57.25. Click **Compute**. The entry field for **a_Persons** with default **2** appears.

57.26. Click **Compute**. The combo box for **a_Weeks** (for person 0) appears. The default value of 2 weeks is selected.

57.27. Click **Compute**. The entry field for **a_DOB** (for person 0) appears.

57.28. Enter **1.1.1950**.

57.29. Click **Compute**. The combo box for **a_CoverageLevel** (for person 0) with default **10K** appears.

57.30. Click **Compute**. The combo box for **a_Weeks[1]** (for person 1) appears. The default value of 2 weeks is selected.

57.31. Click **Compute**. The entry field for **a_DOB[1]** (for person 1) appears.

57.32. Enter **1.1.1800**.

57.33. Click **Compute**. The combo box for **a_CoverageLevel[1]** (for person 1) with default **10K** appears.

57.34. Click **Compute**. The result of **1012** is displayed:

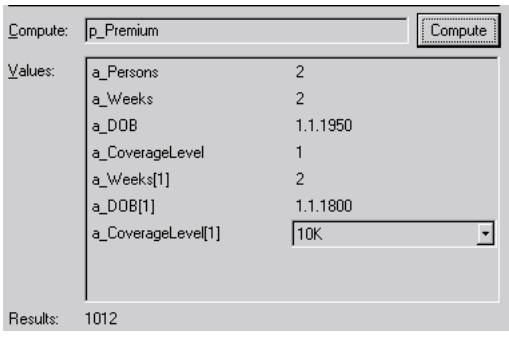


Figure 57.4. Test dialog for multiple inclusion for Product1, Product2

The result of 1012 for p_Premium includes the following:

- Product1 for person 0 = 4 = 1 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- Product1 for person 1 = 8 = 2 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- Product2 for person 0 = 0
- Product2 for person 1 = 1000

57.35. Close the test.

58. Add multiply inclusive products in layout (D)

58.1. Change title of workarea Personal info

58.1. Change property **Title** of workarea **Personal info** to **General info**.

58.2. Delete Product info / All products components

58.2. In workarea **Product info** page **All products**: Delete **radio group** for **a_ProductType**.
58.3. In workarea **Product info** page **All products**: Delete **radio button** for **a_P1**.
58.4. In workarea **Product info** page **All products**: Delete **checkbox** for **a_P2**.

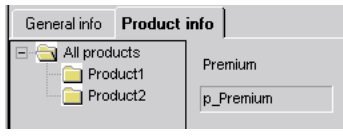


Figure 58.1. Workarea Product info page All products after components deleted

58.3. Create Product1 subnode Product1P

58.5. Right-click on **Product1**. A pop-up dialog appears.
58.6. Select **Insert subnode**. A subnode for Product1 appears.
58.7. Change **Title** to **Product1P**.
58.8. Set **Dynamic** to **Yes**.
58.9. Set **Count Attribute** to **a_Persons**.
58.10. Set **Caption New** to **new1P**.
58.11. Set **Caption Delete** to **delete1P**.
58.12. Set **Data indicator** to **On**.
58.13. Set **DI Rule** to **DI_Product1()** (no "i" in the paratheses).

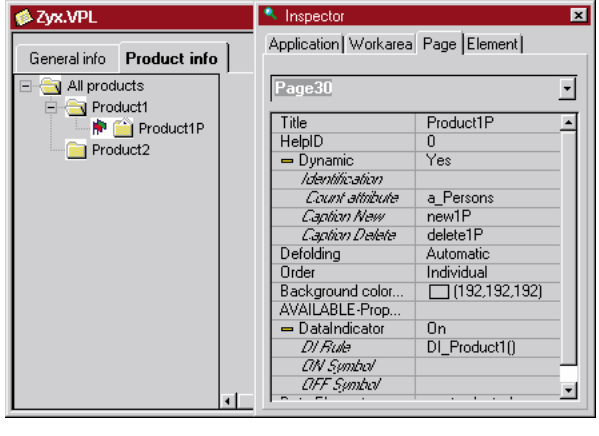


Figure 58.2. Page Product1P settings

58.4. Copy components from page Product1 to Product1P

58.14. In page **Product1**: Delete radio group **a_Weeks**.
58.15. Select (by clicking while holding down the **CTRL** key) all components.
58.16. Right-click. A pop-up dialog appears.
58.17. Select **Cut**. The selected items are cut and pasted to the clipboard.
58.18. Select page **Product1P**.
58.19. Right-click anywhere on the page area. A pop-up dialog appears.
58.20. Select **Paste**. The components are pasted to the page.

58.5. Page Product1: Delete components; change result field source

58.21. In page **Product1**: Delete the following components:
58.21.1. Label **Weeks of coverage**

58.21.2. Drop-down list **a_Weeks**

58.21.3. Radio group for **a_CoverageLevel**

58.22. Change label **Premium** to **Product1 premium**.

58.23. Change the **Source** for the result field to **p_PremiumP1**.

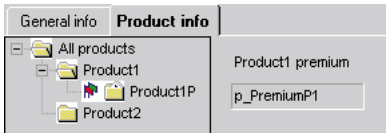


Figure 58.3. Page Product1 components

58.6. Page Product1P: Add a_Name label / entry field; Set identification

58.24. In page **Product1P**: Add a **label** with text **Name**.

58.25. Add an **entry field** with **Source** as **a_Name**.

58.26. Change the name of the element to **ElementName**.

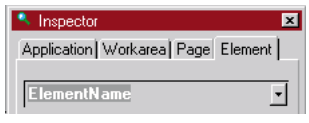


Figure 58.4. Name of a_Name entry field

58.27. Select **Page**.

58.28. Set **Identification** to **ElementName** (note: simply type in if not available from the drop-down list).

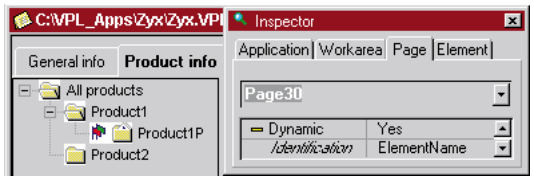


Figure 58.5. Page Product1P identification

58.7. Page Product1P: Modify for premium

58.29. Change **label Product1** to **Product1 for this person**.

58.30. Change **Source** for **result field** to **p_PremiumEach()**.

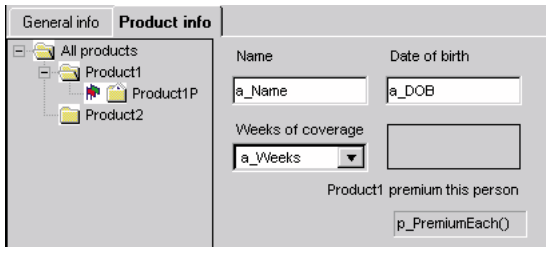


Figure 58.6. Page Product1P components

58.8. Page Product2: Change result field source

58.31. Change label **Premium** to **Product2 premium**.

58.32. Change the **Source** for the result field to **p_PremiumP2**.

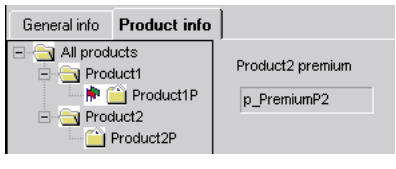


Figure 58.7. Page Product2 components

58.9. Create Product2 subnode Product2P

58.33. Right-click on **Product2**. A pop-up dialog appears.

58.34. Select **Insert subnode**. A subnode for Product2 appears.
58.35. Change **Title** to **Product2P**.
58.36. Set **Dynamic** to **Yes**.
58.37. Set **Count Attribute** to **a_Persons**.
58.38. Set **Caption New** to **new2P**.
58.39. Set **Caption Delete** to **delete2P**.
58.40. Set **Data indicator** to **On**.
58.41. Set **DI Rule** to **DI_Product2()** (no "i" in the paratheses).

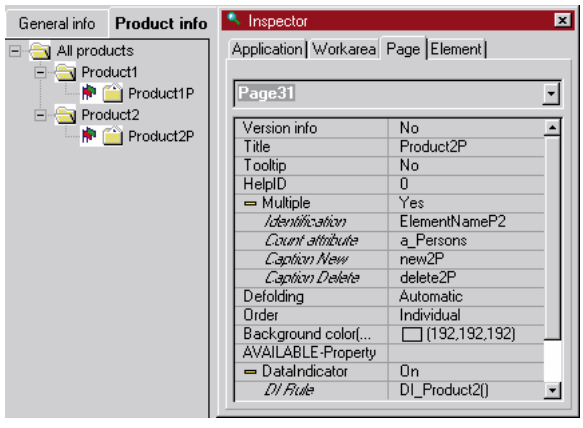


Figure 58.8. Page Product2P settings

58.10. Copy components from page Product1P to Product2P and modify; set page identification

58.42. Copy the following components from **Product1P** to **Product2P**:
58.42.1. Label **Name**
58.42.2. Entry field **a_Name**
58.42.3. Label **Product1 premium this person**
58.42.4. Result field **p_PremiumEach()**
58.43. Change the **Element name** for the entry field to **ElementNameP2**.
58.44. Change the label to **Product2 premium this person**
58.45. Change the result field **Source** to **p_PremiumP2Each()**

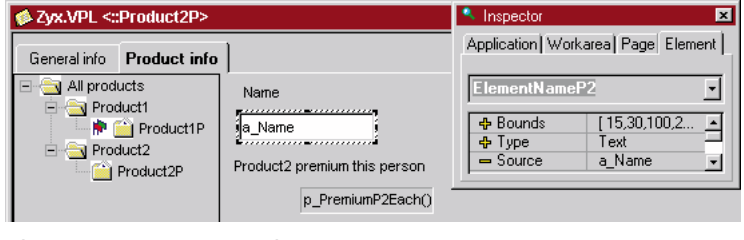


Figure 58.9. Page Product2P components

58.46. Select **Page**.
58.47. Set Identification to **ElementNameP2** (note: simply type in if not available from the drop-down list).

58.11. Move a_DOB label / entry field from workarea General info to Page Product2P

58.48. Move the following components from **workarea General info** to **page Product2P**:
58.48.1. Label **a_DOB**
58.48.2. Entry field **a_DOB**

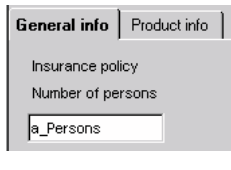


Figure 58.10. Workarea General info components

58.12. Rename pages and components

The pages and components should be renamed to make creating the report easier.

58.49. Rename Workarea **General info** page to **PageGeneralInfo**.
58.50. Rename Workarea **Product info** page for **All products** to **PageAllProducts**.
58.51. Rename Workarea **Product info** page for **Product1** to **PageProduct1**.
58.52. Rename Workarea **Product info** page for **Product1P** to **PageProduct1P**.
58.53. Rename Workarea **Product info** page for **Product2** to **PageProduct2**.
58.54. Rename Workarea **Product info** page for **Product2P** to **PageProduct2P**.
58.55. Rename page **PageGeneralInfo** entry field for **a_Persons** to **Page1NumberPersons**.
58.56. Rename page **PageAllProducts** result field for **p_Premium** to **PageAllProductsPremium**.
58.57. Rename page **PageProduct1** result field for **p_PremiumP1** to **PageProduct1Premium**.
58.58. Rename page **PageProduct1P** entry field for **a_Name** to **PageProduct1PName**.
58.59. Rename page **PageProduct1P** drop-down list for **a_Weeks** to **PageProduct1PWeeks**.
58.60. Rename page **PageProduct1P** radio group for **a_CoverageLevel** to **PageProduct1PCoverageLevel**.
58.61. Rename page **PageProduct1P** result field for **p_PremiumEach()** to **PageProduct1PPremium**.
58.62. Rename page **PageProduct2** result field for **p_PremiumP2** to **PageProduct2Premium**.
58.63. Rename page **PageProduct2P** entry field for **a_Name** to **PageProduct2PName**.
58.64. Rename page **PageProduct2P** entry field for **a_DOB** to **PageProduct2PDOB**.
58.65. Rename page **PageProduct2P** result field for **p_PremiumP2Each()** to **PageProduct2PPremium**.
58.66. Save the layout (Zyx51.vpl).

58.13. Test (Designer)

58.67. Click on **Test Application** (save application). The workarea **General info** appears with default 2 persons.
58.68. Select workarea **Product info**. Note the default persons already created:

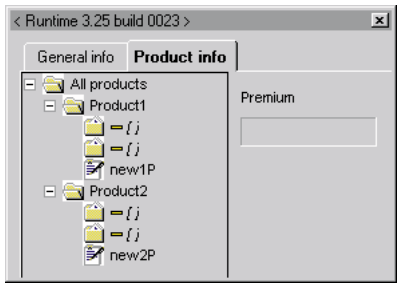


Figure 58.11. Workarea Product info with 2 default persons

58.13.1. Enter required info for Product1 / Person1

58.69. Select the first subnode under **Product1**.
58.70. Enter for **Name**: **Name1**.
58.71. Click to change the focus. Note that the name appears in the page tree.

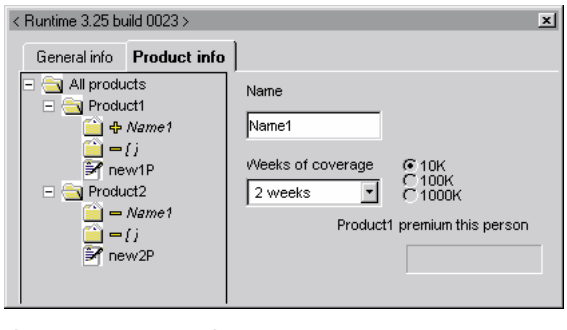


Figure 58.12. Product1 Person1 page

58.13.2. Enter required info for Product1 / Person2

58.72. Select the second subnode under **Product1**.

58.73. Enter for **Name**: **Name2**.

58.13.3. Enter required info for Product2 / Person1

58.74. Select **Name1** subnode under **Product2**.

58.75. Enter for **Date of birth**: **1.1.1900**. Note that the Product2 premium of 0 is displayed.

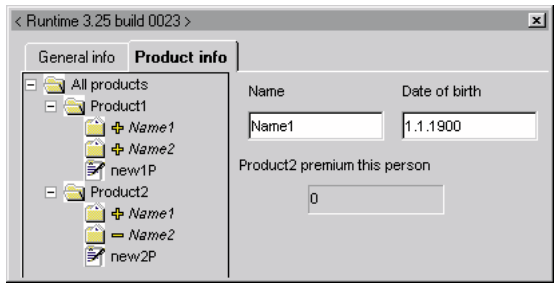


Figure 58.13. Product2 Person1 page

58.13.4. Enter required info for Product2 / Person2

58.76. Select the **Name2** subnode under **Product2**.

58.77. Enter for **Date of birth**: **1.1.1888**. Note that the Product2 premium of 1000 is displayed.

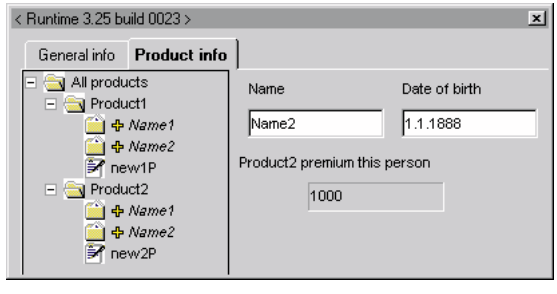


Figure 58.14. Product2 Person1 page

58.13.5. View premium values for All products, Product1, Product2

58.78. Select the **All products, Product1,** and **Product2** pages to display the Premium for each:

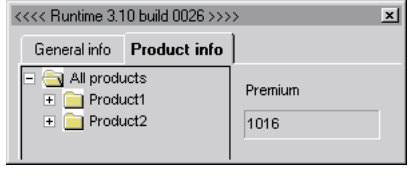


Figure 58.15. Premium in page All products

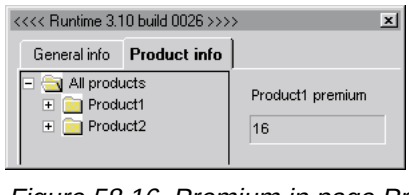


Figure 58.16. Premium in page Product1

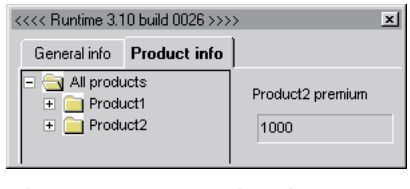


Figure 58.17. Premium in page Product2

58.13.6. Create Person3

In node **Product1**: Click on **new1p**. Note that a new person is created.

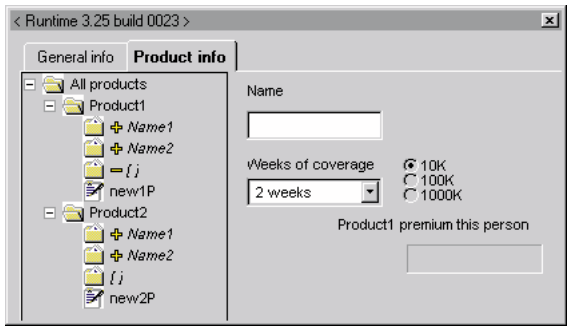


Figure 58.18. New person for Product1

58.79. Enter **Name**: **Name3**.

58.80. Click to change the focus. Note that the Product2 / Name3 page is opened:

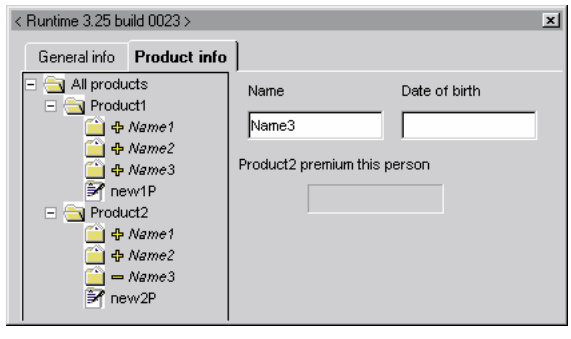


Figure 58.19. Product2 / Name3 page

58.13.7. Create Person4, Person5 (Person6 cannot be created)

58.81. Double-click on **new1p**. A new person is created.

58.82. Double-click on **new1p**. A new person is created.

58.83. Double-click on **new1p**. No new person is created, since the maximum allowed number of persons is 5.

58.84. Close the test dialog.

59. Add multiply inclusive products in layout as a Grid list (W, D)

59.1. Add Product2 properties (W)

59.1. In the **Workbench**: For **Product2**: Create property **Folge(i)** with the following definition

```
i+1
```

59.2. Create property **Name(i)** with the following definition:

```
a_Name[i]
```

59.3. Create property **DOB(i)** with the following definition:

```
a_DOB[i]
```

59.2. Create attribute Start (W)

59.4. Create attribute **Start** with property **default** with value **0**.

59.5. Create attribute **Start** with property **visible** with value **1**.

59.6. Save the model (Zyx52.pms).

59.3. Add grid list to page PageProduct2 (D)

59.7. Add a **Grid List** to page **PageProduct2**.

59.8. Rename to **P2Grid**.

59.9. Set **Column** to **5**.

59.10. Set **Key Column** to **5**.

59.11. Set **Iteration type** to **Attribute**.

59.12. Set **Iteration property** to **a_Persons**.

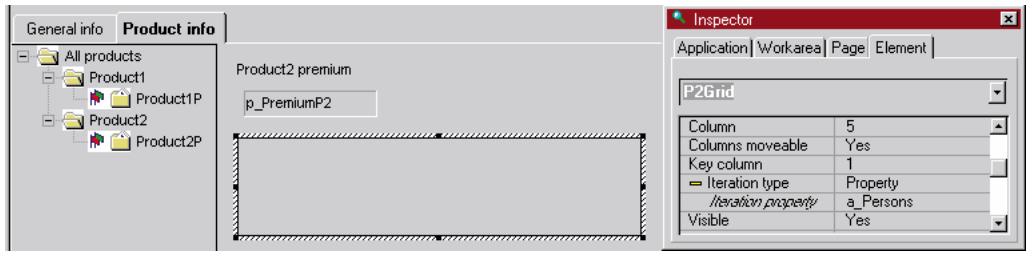


Figure 59.1. Grid list properties

59.4. Specify column 1 settings (D)

59.13. From the **Property for** drop-down list: Select **Column 1**.

59.14. Set **Title** to **Nr**.

59.15. Set **Width** to **40**.

59.16. Set **Usage** to **Output**.

59.17. Set **Property** to **Folge()**.

59.18. Set **Type** to **Number**.

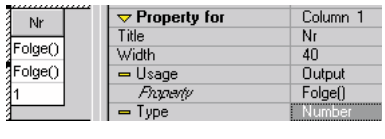


Figure 59.2. Column 1 settings

59.5. Specify column 2 settings

59.19. From the **Property for** drop-down list: Select **Column 2**.

59.20. Set **Title** to **Name**.

59.21. Set **Width** to **70**.

59.22. Set **Usage** to **Output**.

59.23. Set **Property** to **Name()**.

59.24. Set **Type** to **Text**.

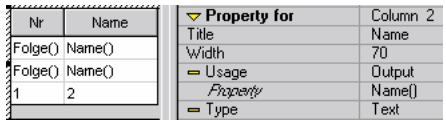


Figure 59.3. Column 2 settings

59.6. Specify column 3 settings

59.25. From the **Property for** drop-down list: Select **Column 3**.
59.26. Set **Title** to **Date of birth**.
59.27. Set **Width** to **70**.
59.28. Set **Usage** to **Output**.
59.29. Set **Property** to **DOB()**.
59.30. Set **Type** to **Date**.
59.31. Set **Format** to **dd/mm/yy**.

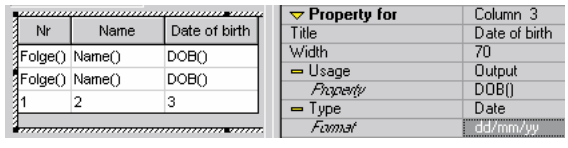


Figure 59.4. Column 3 settings

59.7. Specify column 4 settings

59.32. From the **Property for** drop-down list: Select **Column 4**.
59.33. Set **Title** to **Premium**.
59.34. Set **Width** to **100**.
59.35. Set **Usage** to **Output**.
59.36. Set **Property** to **p_PremiumP2Each()**.
59.37. Set **Type** to **Currency**.

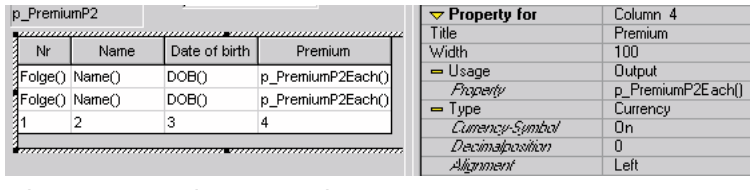


Figure 59.5. Column 4 settings

59.8. Specify column 5 settings

59.38. From the **Property for** drop-down list: Select **Column 5**.
59.39. Set **Title** to **Folge**.
59.40. Set **Width** to **50**.
59.41. Set **Usage** to **Input**.
59.42. Set **Attribute** to **Start**.

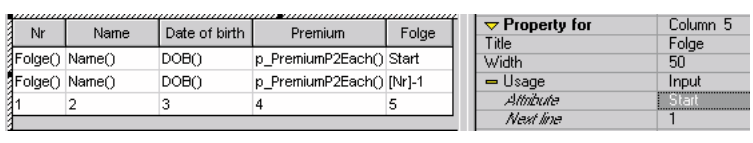


Figure 59.6. Column 5 settings

59.9. Add edit field for Start

59.43. Add an **Edit field**.
59.44. Set the name of the element to **StartE**.
59.45. Set **Source** to **Start**.
59.46. Set **Compute** to **On change**.

59.47. Save the layout (Zyx52.vpl).

59.10. Test (D)

59.48. Click on the **Test** button.

59.10.1. Enter information for 2 persons

59.49. For the first **Person**:

59.49.1. Set **Name** to **Name1**.

59.49.2. Set **Date of birth** to **1.1.1955**.

59.50. For the second **Person**:

59.50.1. Set **Name** to **Name2**.

59.50.2. Set **Date of birth** to **1.1.1850**.

59.51. Select page **Product2**. Note the contents of the grid:

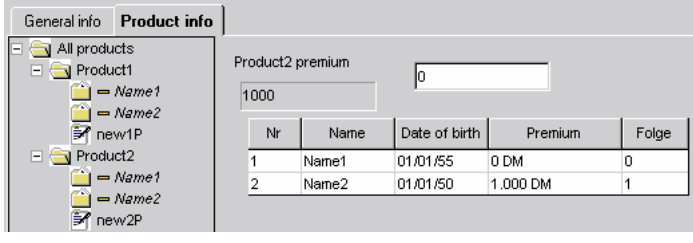


Figure 59.7. Product2 page with 2 persons in Grid

59.10.2. Create 3rd person

59.52. Click on **new2p**. A new person is created.

59.53. For the first **Person**:

59.53.1. Set **Name** to **Name3**.

59.53.2. Set **Date of birth** to **1.1.1811**.

59.54. Select page **Product2**. Note the contents of the grid:

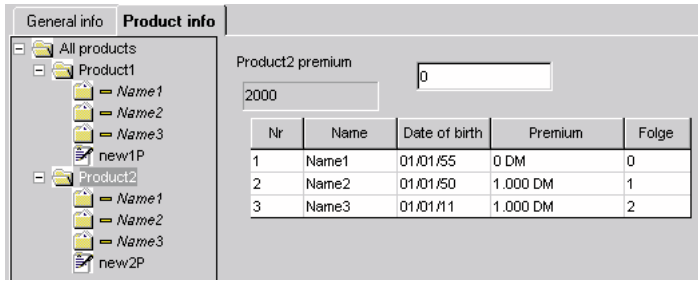


Figure 59.8. Product2 page with 3 persons in Grid

59.55. Close the test dialog.

59.11. Hide Start entry field and Folge grid column (D)

59.56. On page **Product2**: For **entry field Start**: Set **Visible** to **No**.

59.57. For **grid P2Grid**: For **column 5**: Set **Width** to **0**.

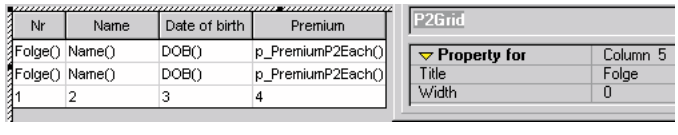


Figure 59.9. Grid with Folge column 5 width set to 0

59.58. Save the layout.

59.12. Test (D)

59.59. Test as previously. Note that the functionality has not changed.

60. Add multiply inclusive products in layout as an Extended Grid list (D)

60.1. Add extended grid list to page PageProduct2

60.1. Add an **Extended Grid List** to page **PageProduct2**.
60.2. Rename to **P2ExtGrid**.
60.3. Set **Column** to **4**.
60.4. Set **Count attribute** to **a_Persons**.
60.5. Set **Caption New** to **new person**.
60.6. Set **Caption Delete** to **delete person**.

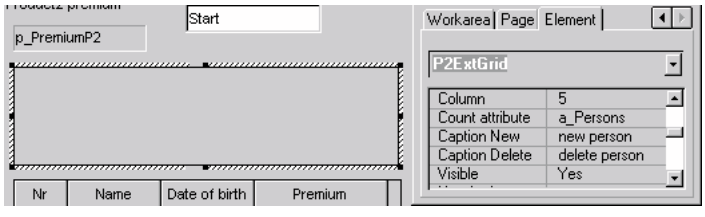


Figure 60.1. Extended Grid list properties

60.2. Specify column 1 settings

60.7. From the **Property for** drop-down list: Select **Column 1**.
60.8. Set **Title** to **Nr**.
60.9. Set **Width** to **40**.
60.10. Set **Usage** to **Output**.
60.11. Set **Type** to **Number**.
60.12. Set **Property** to **Folge()**.

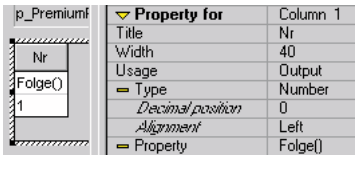


Figure 60.2. Column 1 settings

60.3. Specify column 2 settings

60.13. From the **Property for** drop-down list: Select **Column 2**.
60.14. Set **Title** to **Name**.
60.15. Set **Width** to **70**.
60.16. Set **Usage** to **Input**.
60.17. Set **Type** to **Text**.
60.18. Set **Attribute** to **a_Name()**.

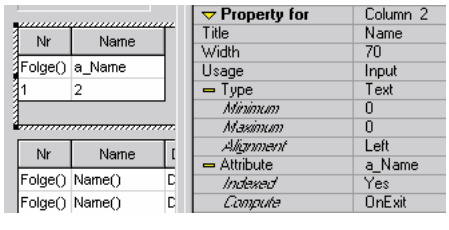


Figure 60.3. Column 2 settings

60.4. Specify column 3 settings

60.19. From the **Property for** drop-down list: Select **Column 3**.
60.20. Set **Title** to **Date of birth**.
60.21. Set **Width** to **70**.

60.22. Set **Usage** to **Input**.
60.23. Set **Type** to **Date**.
60.24. Set **Format** to **dd.mm.yyyy**.
60.25. Set **Attribute** to **a_DOB**.

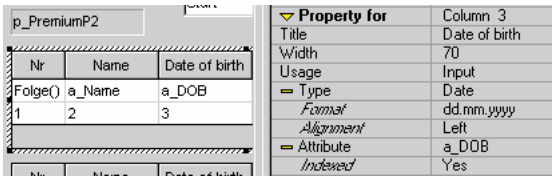


Figure 60.4. Column 3 settings

60.5. Specify column 4 settings

60.26. From the **Property for** drop-down list: Select **Column 4**.
60.27. Set **Title** to **Premium**.
60.28. Set **Width** to **100**.
60.29. Set **Usage** to **Output**.
60.30. Set **Type** to **Currency**.
60.31. Set **Property** to **p_PremiumP2Each()**.

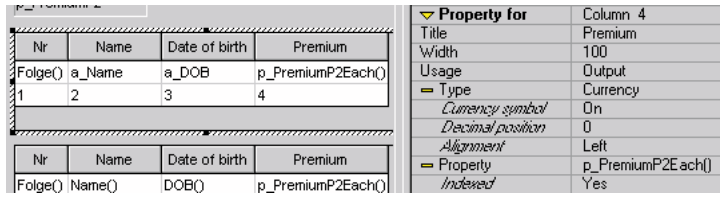


Figure 60.5. Column 4 settings

60.32. Save the layout (Zyx53.vpl).

60.6. Test (D)

60.33. Click on the **Test** button.

60.6.1. Enter information for first person in the extended grid list

60.34. In the **Extended grid**: Click in column **Name** row **1**.
60.35. Enter **Name1**.
60.36. Click in column **Date of birth** row **1**. Note that the change of focus causes the entered data to be displayed in the regular grid list.

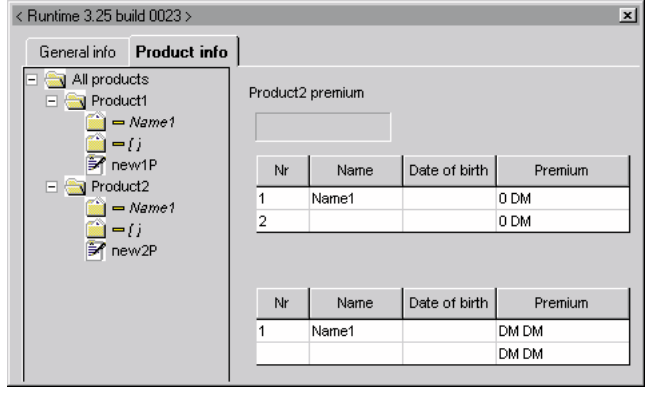


Figure 60.6. Product2 page Extended Grid (above) and regular Grid (below)

60.37. In column **Date of birth** row **1**: Enter **01.01.1800**.
60.38. Click in column **Premium** row **1**. The page changes.

60.39. Click on page **Product2**. Note that the date and the premium are displayed in both grids.

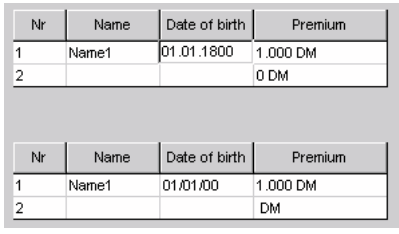


Figure 60.7. Date of birth and premium in Extended Grid (above) and regular Grid (below)

60.6.2. Add new person to list

60.40. In the **Extended grid**: Right-click. A pop-up window appears:

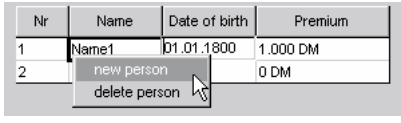


Figure 60.8. Popup window for Extended Grid

60.41. Click on **new person**. A new person appears in the extended grid:

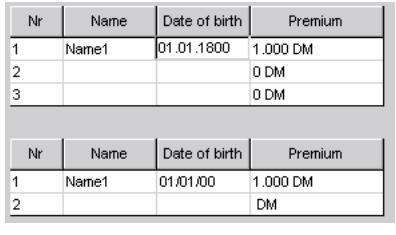


Figure 60.9. New person in Extended Grid

60.42. For 2nd person: Enter **Name** as **Name2**.
60.43. For 2nd person: Enter **Date of birth** as **01.01.1955**.
60.44. For 3rd person: Enter **Name** as **Name3**.
60.45. For 3rd person: Enter **Date of birth** as **01.01.1800**.
60.46. Change the focus. Note that the new person is not displayed in the tree.

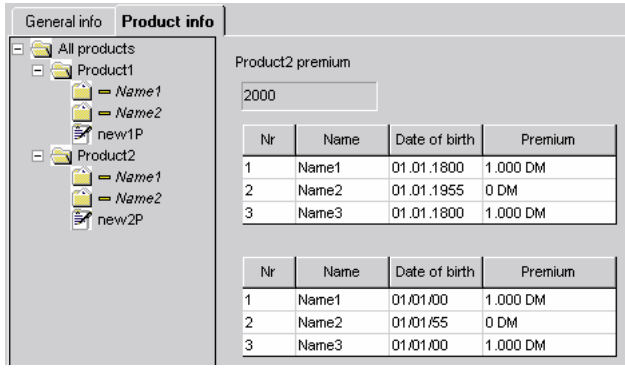


Figure 60.10. New person is not in the tree

60.47. Double-click on **new2P**. Note that the new person is displayed in the tree.

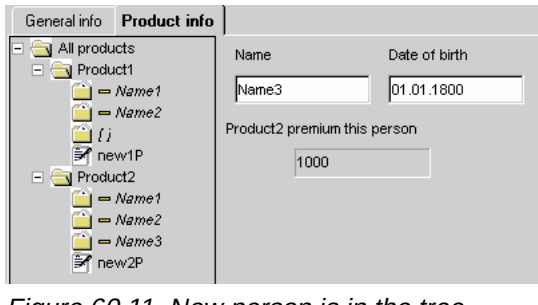


Figure 60.11. New person is in the tree

60.6.3. Delete person from list

60.48. In the **Extended grid**: Right-click in the row for **Name3**. A pop-up window appears.
60.49. Click **delete person**.
60.50. Select a different page.
60.51. Select page **Product2**. Note that the person has been deleted:

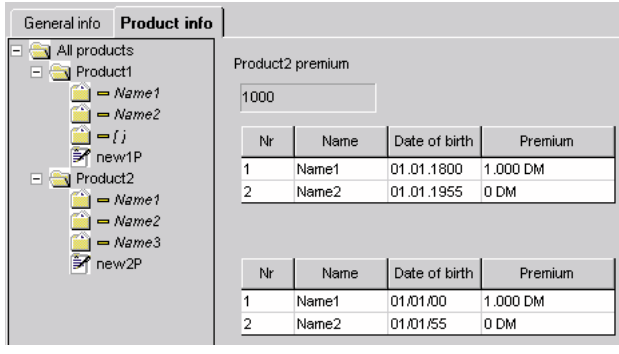


Figure 60.12. Name3 deleted from Extended Grid

60.52. Delete **Name3** manually from the tree.

61. Add multiply inclusive products in layout as a Graphics element (D)

61.1. Add graphics element to page PageProduct2

61.1. Add a **Graphics element** to page **PageProduct2**.
61.2. Rename to **P2Graphics**.
61.3. Set **Identification grid** to **P2ExtGrid**.
61.4. Set **X-axe name** to **Person number**.
61.5. Set **Y-axe name** to **Premium**.
61.6. Set **Key column** to **1**.
61.7. Set **Date column** to **4**.

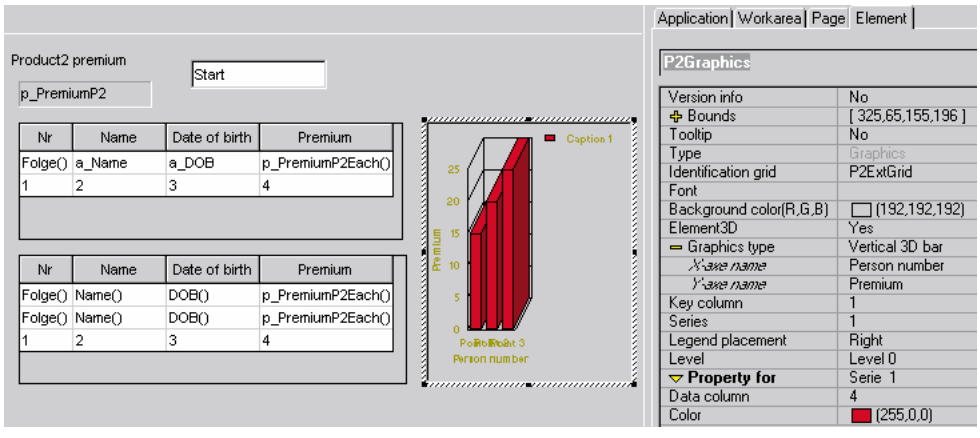


Figure 61.1. Graphics element properties

61.8. Save the layout (Zyx54.vpl).

61.2. Test (D)

61.9. Click on the **Test** button.

61.2.1. Enter information for first person in the extended grid list

61.10. In the **Extended grid**: For the 1st person: Enter **Name** as **Name1**.
61.11. For the 1st person: Enter **Date of birth** as **01.01.1800**.
61.12. Click to change the focus.
61.13. Select page **Product2**. Note that the premium is displayed for **Name1**:

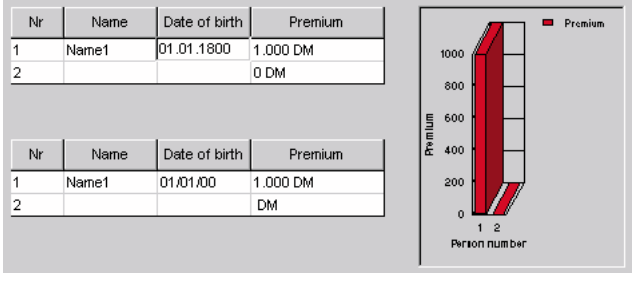


Figure 61.2. Product2 info displayed in graphics element

62. Add multiply inclusive products in Report (RE)

62.1. Modify main text / main bereich

62.1. Replace the following text:

```
Personal Info
x_persons persons.
Date of birth (for all persons) is x_dob.
Age premium is x_agepremium.
[pagebreak]
All products
Premium is x_cost.
```

with the following text:

```
General Info [ZyxHeading1 paragraph style]
x_persons persons.
[pagebreak]
Premium totals [ZyxHeading1 paragraph style]
Premium is x_cost.
Product1 premium is x_cost1.
Product2 premium is x_cost2.
```

62.2. Specify the above text in bold as datafields.

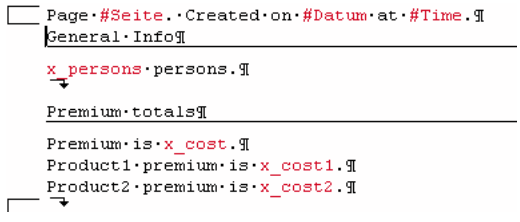


Figure 62.1. New text for general area

62.3. Change the bereichskommando for the document to the following:

```
VPMSName = 'C:\VPL_Apps\Zyx\Zyx.vpm';
VPMSTRANS = 'C:\VPL_Apps\Zyx\Zyx.vpd';
Steuerdatei $DATEI_ROOT;
x_persons = COMPUTE("a_Persons");
x_cost = PageAllProductsPremium;
x_cost1 = PageProduct1Premium;
x_cost2 = PageProduct2Premium;
```

62.2. Modify Product1 text / bereich

62.4. Replace the following text:

```
[pagebreak]
Product1
Number of weeks is x_weeks.
Coverage level is x_coveragelevel.
```

with the following text:

```
[pagebreak]
Product1 for x_name. [ZyxHeading1 paragraph style]
Number of weeks is x_weeks.
Coverage level is x_coveragelevel.
Premium is x_cost.
```

62.5. Specify the above text in bold as datafields.

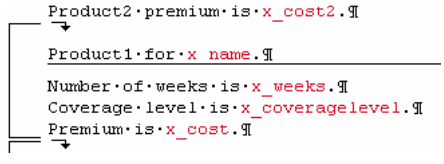


Figure 62.2. New text for Product1 area

62.6. Change the bereichskommando for the document to the following:

```
Steuerdatei $Datei_PageProduct1P ;
x_name = PageProduct1PName;
i=instr(PageProduct1PWeeks,"#");
x_weeks = mid$(PageProduct1PWeeks,i+1,7);
i=instr(PageProduct1PCoverageLevel,"#");
x_coveragelevel = mid$(PageProduct1PCoverageLevel,i+1,5);
x_cost = PageProduct1PPremium;
```

62.3. Modify Product2 text / bereich

62.7. Replace the following text:

```
[pagebreak]
Product2
(no input required for Product2).
```

with the following text:

```
[pagebreak]
Product2 for x_name.
Date of birth is x_dob.
Age premium is x_agepremium.
```

62.8. Specify the above text in bold as datafields.

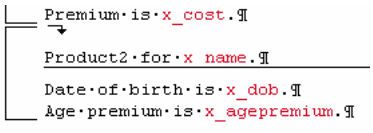


Figure 62.3. New text for Product2 area

62.9. Change the bereichskommando for the document to the following:

```
Steuerdatei $Datei_PageProduct2P ;
x_name = PageProduct2PName;
x_dob = PageProduct2PDOB;
x_agepremium = PageProduct2PPremium;
```

62.10. Save the report (Zyx55.cat).

62.4. Test (VF)

62.11. Restart **VFrame**.

62.12. Select the **Zyx** consultation.

62.13. In the **Product info** tab: For **Product1** first person: Enter **Name** as **name1**.

62.14. For **Product1** first person: Select **Weeks of coverage** as **1 week**.

62.15. For **Product1** second person: Enter **Name** as **name2**.

62.16. For **Product2 name1**: Enter **Date of birth** as **1.1.1900**.

62.17. For **Product2 name2**: Enter **Date of birth** as **1.1.1800**.

62.18. Print a report. The report contains the following pages:

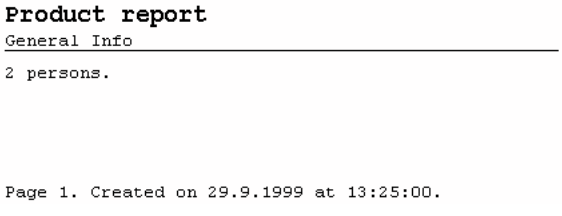


Figure 62.4. Printout for multiple inclusion page 1

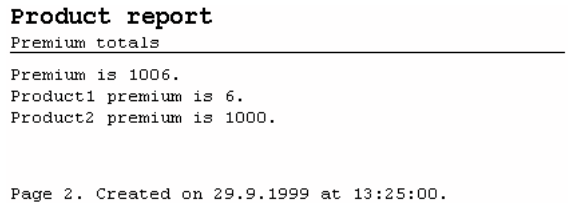


Figure 62.5. Printout for multiple inclusion page 2

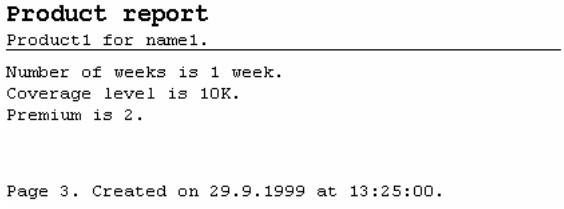


Figure 62.6. Printout for multiple inclusion page 3

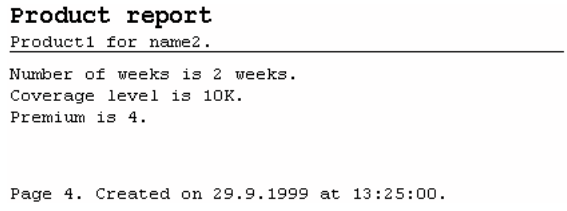


Figure 62.7. Printout for multiple inclusion page 4

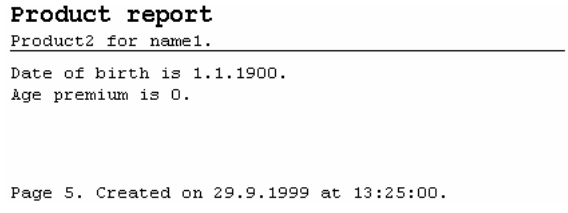


Figure 62.8. Printout for multiple inclusion page 5

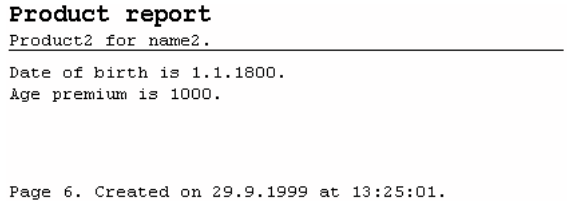


Figure 62.9. Printout for multiple inclusion page 6

63. Add Objects to the product tree (W)

63.1. Delete properties in ProductMain, Product1, Product2

63.1. For **ProductMain**: Delete property **p_Premium**.
63.2. For **Product1**: Delete property **p_PremiumP1**.
63.3. For **Product1**: Delete property **p_PremiumEach(i)**.
63.4. For **Product1**: Delete property **DI_Product(i)**.
63.5. For **Product2**: Delete property **p_PremiumP2)**.
63.6. For **Product2**: Delete property **p_PremiumP2Each(i)**.

63.2. Set Product1, Product2 inclusion to mandatory

63.7. For **Product1**: Set **IncType** to **mandatory**.
63.8. For **Product1**: Set **IncRule** to blank.
63.9. For **Product2**: Set **IncType** to **mandatory**.
63.10. For **Product2**: Set **IncRule** to blank.

63.3. Create Object1

63.11. In the **Objects** window: Right-click.
63.12. Select **New**. A new object is created.
63.13. Type the following: **Object1**.

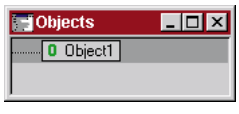


Figure 63.1. Object1 in the Objects window

63.4. Set Object1 properties in window Objects

63.14. Double-click on **Object1**.
63.15. Add property **DI_Object1(i)** with the following definition:

```
if(i=#a_Persons;
        if(
        not(undefined("a_DOB" & "[" & #a_Persons & "]"))
        &&
        days(a_DOB[#a_Persons]) >0
        &&
        not(undefined("a_Name" & "[" & #a_Persons & "]"))
        &&
        length(a_Name[#a_Persons]) > 0
        &&
        not(undefined("a_CoverageLevel" & "[" & #a_Persons & "]"))
        &&
        a_CoverageLevel[#a_Persons] >0 ;
        1;
        0);
  0)
```

63.5. Add Object1 to Product1 product tree (with Drag&Drop)

63.16. Left-click without releasing the mouse-button on **Object1**.
63.17. Move the mouse until the mouse pointer is over **Product1**. Note that the mouse cursor is a down-

ward-pointing arrow.

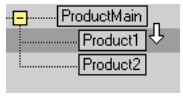


Figure 63.2. Adding Object1 to the Product1 product tree

63.18. Release the mouse button. Note that Object1 has now been added to the Product1 product tree.

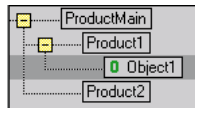


Figure 63.3. Object1 in the Product1 product tree

63.6. Specify multiple inclusion for Object1

63.19. Double-click on **Object1** in the **Products** window. Note that the property definitions are not displayed.

63.20. Set **IncType** to **multiple**.

63.21. Set **IncRule** to **a_Persons**.

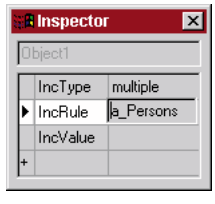


Figure 63.4. Multiple inclusion specification for Object1

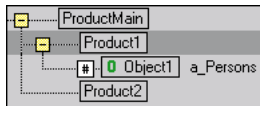


Figure 63.5. Object1 in the products window with multiple inclusion

63.7. Add Object1 to Product2 product tree

63.22. Add **Object2** as a subnode of **Product2** (with Drag&Drop).

63.8. Specify multiple inclusion for Object2

63.23. Double-click on **Object2** in the **Products** window.

63.24. Set **IncType** to **multiple**.

63.25. Set **IncRule** to **a_Persons**.

63.9. Set Product1 / Object1 properties in window Products

63.26. In the **Products** window: Double-click on **Object1** under **Product1**.

63.27. Add property **p_Premium** with the following definition:

```
a_Weeks * f_AgePremium * f_CoverageLevelPremium
```

63.28. Add property **p_PremiumEach(i)** with the following definition:

```
if(i=#a_Persons;
     p_Premium;
     0)
```

63.10. Set Product2 / Object1 properties in window Products

63.29. In the **Products** window: Double-click on **Object1** under **Product2**.

63.30. Add property **p_Premium** with the following definition:

```
if (f_Age(a_DOB) > 100;1000;0)
```

63.31. Add property **p_PremiumP2Each(i)** with the following definition:

```
if(i=#a_Persons;
```

```
p_Premium;
0)
```

63.32. Save the model (Zyx56.pms).

63.11. Test

63.33. Click the **Test** button.

63.34. Click **Compute**. Note that the result is the same as in the previous test.

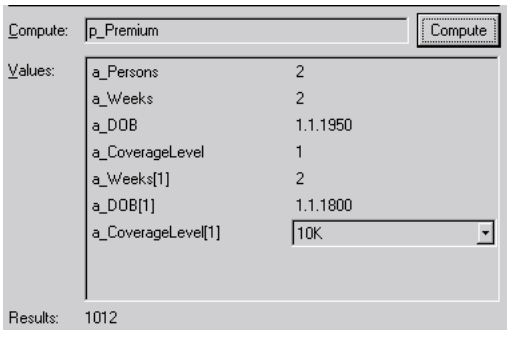


Figure 63.6. Test dialog for multiple inclusion of Object1 under Product1, Product2

The result of 1012 for p_Premium includes the following:

- p_Premium for Product1 / Object1[0] = 4 = 1 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- p_Premium for Product1 / Object1[1] = 8 = 2 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- p_Premium for Product2 / Object1[0] = 0
- p_Premium for Product2 / Object1[1] = 1000

63.35. Close the test.

64. Add Objects to the layout (D)

64.1. Change DI Rule for pages PageProduct1P, PageProduct2P (D)

64.1. Change **DI Rule** for page **PageProduct1P** to **DI_Object1()**.

64.2. Change **DI Rule** for page **PageProduct2P** to **DI_Object1()**.

64.3. Save the layout (Zyx57.vpl).

64.2. Test (D)

64.4. Test in the Designer. Note that the results are the same as in the previous Designer test except for:

- An extra premium of 100 for each product/person
- The data indicator for both product trees turns into a plus sign when all required data for the object has been entered.

64.3. Test (VF)

64.5. Test in VFrame. Note that the results are the same as in the Designer.

64.4. Test (RV)

64.6. Print from VFrame. Note that the results are the same as in the previous print test.

65. Add Events to the product tree (W)

65.1. Delete property p_Premium for Product2 / Object1

65.1. Delete property **p_Premium** for **Object1** under **Product2**.

65.2. Create Event1, Event2; add under Product2 / Object1with multiple inclusion

65.2. Create event **Event1**.
65.3. Drag **Event1** to **Product2 / Object1**.
65.4. For **Product2 / Object1 / Event1**: Set property **IncType** to **multiple**.
65.5. For **Product2 / Object1 / Event1**: Set property **IncRule** to **a_Event1s**.
65.6. Create event **Event2**.
65.7. Drag **Event2** to **Product2 / Object1**.
65.8. For **Product2 / Object1 / Event2**: Set property **IncType** to **multiple**.
65.9. For **Product2 / Object1 / Event2**: Set property **IncRule** to **a_Event2s**.

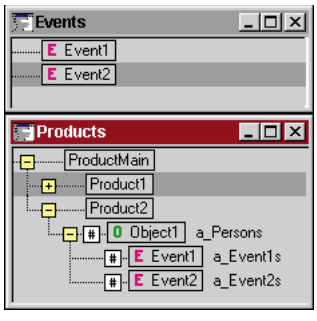


Figure 65.1. Product2 / Object1 / Event1, Event2

65.3. Add p_Premium, p_PremiumEach properties for Events

65.10. For **Product2 / Object1 / Event1**: Add property **p_Premium** with value **10000**.
65.11. For **Product2 / Object1 / Event1**: Add property **p_PremiumP2O1E1Each(i,j)** with the following definition:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      p_Premium;
      0);
   0)
```

65.12. For **Product2 / Object1 / Event2**: Add property **p_Premium** with value **100000**.
65.13. For **Product2 / Object1 / Event2**: Add property **p_PremiumP2O1E2Each(i,j)** with the following definition:

```
if( i=#a_Persons;
   if (j=#a_Event2s;
      p_Premium;
      0);
   0)
```

65.4. Create attribute a_NameEvent1, a_NameEvent2

65.14. Create attribute **a_NameEvent1**.
65.15. Create attribute **a_NameEvent2**.

65.5. Create attributes a_Event1s, a_Event2s with defaults 2, 3

65.16. Create attribute **a_Event1s**.
65.17. Add property **default** with value **2**.
65.18. Create attribute **a_Event2s**.
65.19. Add property **default** with value **3**.

65.6. Add whichobject properties for Object1, Event1, Event2

65.20. For **Product2 / Object1**: Add property **whichobject(i)** with the following code:

```
if( i=#a_Persons;
   i;
   0)
```

65.21. For **Product2 / Object1 / Event1**: Add property **whichobjectE1(i,j)** with the following code:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      i;
      0);
   0)
```

65.22. For **Product2 / Object1 / Event2**: Add property **whichobjectE2(i,j)** with the following code:

```
if( i=#a_Persons;
   if (j=#a_Event2s;
      i;
      0);
   0)
```

65.23.

65.24. Save the model (Zyx58.pms).

65.25. Create runtime model.

65.7. Test (W)

65.26. Click on the **Test** button. The previous test appears.

65.27. Click **Compute**. The entry field for **a_Event1s** with default **2** appears.

65.28. Click **Compute**. The entry field for **a_Event2s** with default **3** appears.

65.29. Click **Compute**. The result of **640412** is displayed. This result includes the following:

- p_Premium for Product2 / Object1[0] / Event1[0] = 10000
- p_Premium for Product2 / Object1[0] / Event1[1] = 10000
- p_Premium for Product2 / Object1[1] / Event1[0] = 10000
- p_Premium for Product2 / Object1[1] / Event1[1] = 10000
- p_Premium for Product2 / Object1[0] / Event2[0] = 100000
- p_Premium for Product2 / Object1[0] / Event2[1] = 100000
- p_Premium for Product2 / Object1[0] / Event2[2] = 100000
- p_Premium for Product2 / Object1[1] / Event2[0] = 100000
- p_Premium for Product2 / Object1[1] / Event2[1] = 100000
- p_Premium for Product2 / Object1[1] / Event2[2] = 100000

65.30. Close the **Test** window.

66. Add Events to the layout (D)

66.1. Create Product2P subnode Product2PE1

66.1. Right-click on **Product2P**. A pop-up dialog appears.
66.2. Select **Insert subnode**. A subnode for Product2P appears.
66.3. Change **Title** to **Product2PE1**.
66.4. Set **Multiple** to **Yes**.
66.5. Set **Count Attribute** to **a_Event1s**.
66.6. Set **Caption New** to **new event1**.
66.7. Set **Caption Delete** to **delete event1**.

66.2. Add elements to page Product2PE1

66.8. Add element **Label** with name **Event1Name** with **Title** as **Event1 Name**.
66.9. Add element **Entry field** with name **EventName** with **Source** as **a_NameEvent1**.
66.10. For element **Entry field**: Set **Compute** to **On change**.
66.11. Add element **Label** with name **eeeffffff** with **Title** as **Product2 Person Event1 premium**.
66.12. Add element **Result field** with name **pp2o1e1** with **Source** as **p_PremiumP2O1E1Each()**.

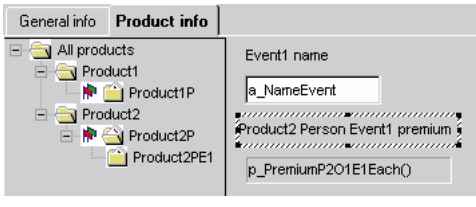


Figure 66.1. Elements for page Product2PE1

66.3. Set Product2P subnode Product2PE1 identification

66.13. Set **Identification** to **EventName**.

66.4. Create Product2P subnode Product2PE2

66.14. Right-click on **Product2P**. A pop-up dialog appears.
66.15. Select **Insert subnode**. A subnode for Product2P appears.
66.16. Change **Title** to **Product2PE2**.
66.17. Set **Multiple** to **Yes**.
66.18. Set **Count Attribute** to **a_Event2s**.
66.19. Set **Caption New** to **new event2**.
66.20. Set **Caption Delete** to **delete event2**.

66.5. Add elements to page Product2PE2

66.21. Add element **Label** with name **xxxdfdfd** with **Title** as **Event2 Name**.
66.22. Add element **Entry field** with name **Event2Name** with **Source** as **a_NameEvent2**.
66.23. For element **Entry field**: Set **Compute** to **On change**.
66.24. Add element **Label** with name **xxxxx** with **Title** as **Product2 Person Event2 premium**.
66.25. Add element **Result field** with name **xxvcvcvcvc** with **Source** as **p_PremiumP2O1E2Each()**.

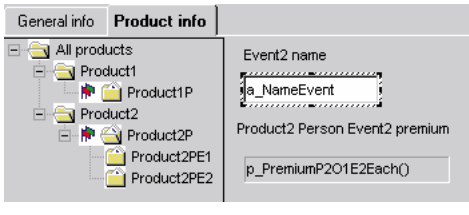


Figure 66.2. Elements for page Product2PE2

66.6. Set Product2P subnode Product2PE2 identification

66.26. Set **Identification** to **Event2Name**.

66.7. Add whichobject result fields to pages Product2P, Product2PE1, Product2PE2

66.27. On page **Product2P**: Add element **Result field** with:

66.27.1. **Name**: **wobj**

66.27.2. **Source**: **whichobject()**

66.27.3. **Visible**: **No**

66.27.4. **Multiple**: **Yes**

66.28. On page **Product2PE1**: Add element **Result field** with:

66.28.1. **Name**: **wobje1**

66.28.2. **Source**: **whichobjectE1()**

66.28.3. **Visible**: **No**

66.28.4. **Multiple**: **Yes**

66.29. On page **Product2PE2**: Add element **Result field** with:

66.29.1. **Name**: **wobje2**

66.29.2. **Source**: **whichobjectE2()**

66.29.3. **Visible**: **No**

66.29.4. **Multiple**: **Yes**

66.30. Save the layout (Zyx59.vpl).

66.8. Test (D)

66.31. Click the **Test** button.

66.32. Select the **Product info** page. Note the Events in the list.

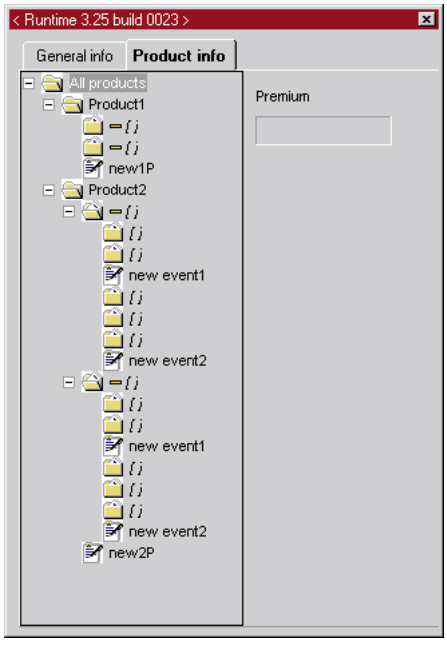


Figure 66.3. Designer test page for the new events

66.33. For all of the objects (persons) and events: Enter names as shown in the following diagram:

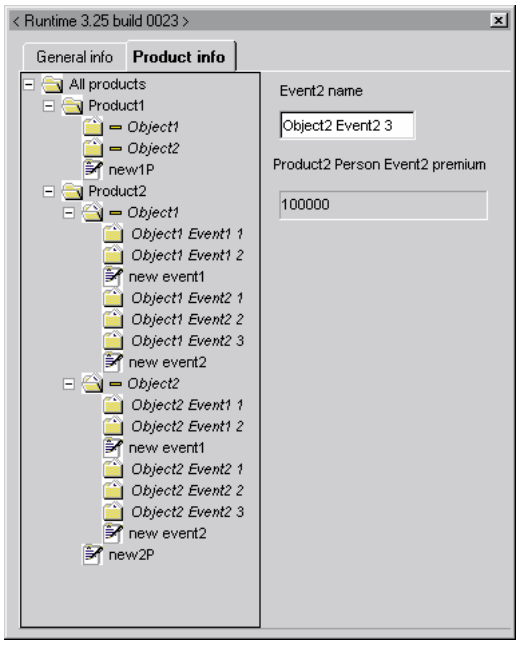


Figure 66.4. Designer test page with names for objects and events entered

66.34. For **Object1**: Enter **Date of birth** as **1.1.1950**.

66.35. For **Object2**: Enter **Date of birth** as **1.1.1850**. Note that the total premium is **640412** (as in the Workbench test).

67. Add Events to Report (RE)

67.1. Add text to report

67.1. Add the following text to the end of the report:

```
Product2 Object x_name2 Event1 x_event.
Event premium is x_eventpremium.
Product2 Object x_name Event2 x_event.
Event premium is x_eventpremium.
```

67.2. Mark the bold text above as datafields.

67.3. Include the above text in the last bereich.

67.2. Create new bereichs for added text

67.4. Create 2 new bereichs for the above text as shown in the following diagram:

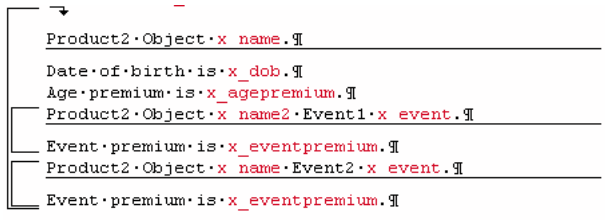


Figure 67.1. New text and new bereichs for events in the report

67.5. Enter the following bereichskommando text for the first sub-bereich:

```
Steuerdatei $Datei_Page42;
if (x_wobj = wobje1);
x_event = EventName;
x_eventpremium = pp2o1e1;
x_name2 = PageProduct2PName;
```

67.6. Enter the following bereichskommando text for the second sub-bereich:

```
Steuerdatei $Datei_Event2Name ;
if (x_wobj = wobje2);
x_event = Event2Name;
x_eventpremium = e334343;
```

67.3. Increase page size of report

67.7. Increase page size of report.

67.8. Save the report (Zyx60.cat).

67.4. Test (VF)

67.9. Open the **Zyx consultation** in VFrame.

67.10. Enter the name of the objects and events as shown in the following diagram.

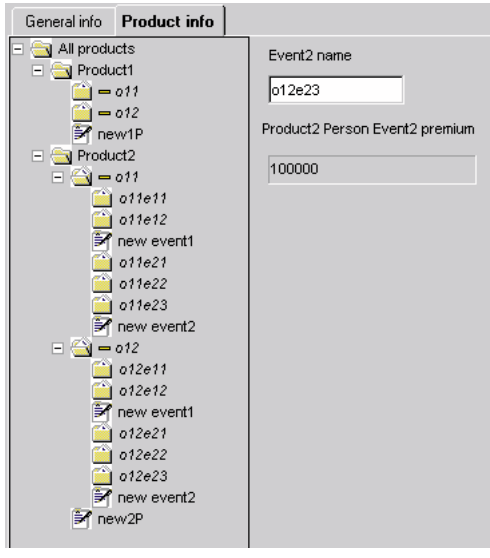


Figure 67.2. Consultation with entered object (person) and event names

67.11. For **o11**: Enter **Date of birth** as **1.1.1950**.
67.12. For **o12**: Enter **Date of birth** as **1.1.1850**. Note that the total premium of 640412 is now displayed.
67.13. Print a report. Note the pages for objects o11 and o12:

Product report

Product2 Object o11.

Date of birth is 1.1.1950.
Age premium is 320000.

Product2 Object o11 Event1 o11e11.

Event premium is 10000.

Product2 Object o11 Event1 o11e12.

Event premium is 10000.

Product2 Object o11 Event2 o11e21.

Event premium is 100000.

Product2 Object o11 Event2 o11e22.

Event premium is 100000.

Product2 Object o11 Event2 o11e23.

Event premium is 100000.

Page 5. Created on 1.10.1999 at 14:15:27.

Figure 67.3. Consultation prinout page for object o11

Product report

Product2 Object o12.

Date of birth is 1.1.1850.
Age premium is 320000.

Product2 Object o12 Event1 o12e11.

Event premium is 10000.

Product2 Object o12 Event1 o12e12.

Event premium is 10000.

Product2 Object o12 Event2 o12e21.

Event premium is 100000.

Product2 Object o12 Event2 o12e22.

Event premium is 100000.

Product2 Object o12 Event2 o12e23.

Event premium is 100000.

Page 6. Created on 1.10.1999 at 14:15:27.

Figure 67.4. Consultation prinout page for object o12

68. Add Compensations to the product tree (W)

68.1. Delete property p_Premium for Product2 / Object1 / Event1

68.1. Delete property **p_Premium** for **Event1** under **Object1** under **Product2**.

68.2. Create Compensation1, Compensation2; add under Product2 / Object1 / Event1 with multiple inclusion

68.2. Create compensation **Compensation1**.
68.3. Drag **Compensation1** to **Product2 / Object1 / Event1**.
68.4. For **Product2 / Object1 / Event1** / **Compensation1**: Set property **IncType** to **multiple**.
68.5. For **Product2 / Object1 / Event1** / **Compensation1**: Set property **IncRule** to **a_Compensation1s**.
68.6. Create compensation **Compensation2**.
68.7. Drag **Compensation2** to **Product2 / Object1 / Event1**.
68.8. For **Product2 / Object1 / Event1** / **Compensation2**: Set property **IncType** to **multiple**.
68.9. For **Product2 / Object1 / Event1** / **Compensation2**: Set property **IncRule** to **a_Compensation2s**.

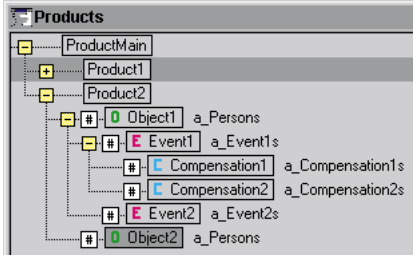


Figure 68.1. Product2 / Object1 / Event1 / Compensation1, Compensation2

68.3. Add p_Premium, p_PremiumEach properties for Compensations

68.10. For **Product2 / Object1 / Event1 / Compensation1**: Add property **p_Premium** with value **1000000**.
68.11. For **Product2 / Object1 / Event1** / **Compensation1**: Add property **p_PremiumP2O1E1C1Each(i,j,k)** with the following definition:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      if (k=#a_Compensation1s;
         p_Premium;
         0);
      0);
   0)
```

68.12. For **Product2 / Object1 / Event1 / Compensation2**: Add property **p_Premium** with value **100000000**.
68.13. For **Product2 / Object1 / Event1** / **Compensation2**: Add property **p_PremiumP2O1E1C2Each(i,j,k)** with the following definition:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      if (k=#a_Compensation2s;
         p_Premium;
         0);
      0);
   0)
```

68.4. Create attribute a_NameCompensation1, a_NameCompensation2

68.14. Create attribute **a_NameCompensation1**.
68.15. Create attribute **a_NameCompensation2**.

68.5. Create attributes a_Compensation1s, a_Compensation2s with defaults 2, 3

68.16. Create attribute **a_Compensation1s**.

68.17. Add property **default** with value **2**.

68.18. Create attribute **a_Compensation2s**.

68.19. Add property **default** with value **3**.

68.6. Add whichevent properties for Event1, Compensation1, Compensation2

68.20. For **Product2 / Object1 / Event1**: Add property **whichevent(i,j)** with the following code:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      j;
      0);
   0)
```

68.21. For **Product2 / Object1 / Event1 / Compensation1**: Add property **whichobjectE1C1(i,j,k)** with the following code:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      if (k=#a_Compensation1s;
         i;
         0);
      0);
   0)
```

68.22. For **Product2 / Object1 / Event1 / Compensation1**: Add property **whicheventC1(i,j,k)** with the following code:

```
if( i=#a_Persons;
   if (j=#a_Event1s;
      if (k=#a_Compensation1s;
         j;
         0);
      0);
   0)
```

68.23. For **Product2 / Object1 / Event1 / Compensation2**: Add property **whicheventC2(i,j)** with the following code:

```
if( i=#a_Event1s;
   if (j=#a_Compensation2s;
      i;
      0);
   0)
```

68.24. Save the model (Zyx61.pms).

68.25. Create runtime model.

68.7. Test

68.26. Click on the **Test** button. The previous test appears.

68.27. Click **Compute**. The entry field for **a_Compensation1s** with default **2** appears.

68.28. Click **Compute**. The entry field for **a_Compensation2s** with default **3** appears.

68.29. Click **Compute**. The result of **1 208 600 412** is displayed. This result includes the following:

- p_Premium for Product2 / Object1[0] / Event1[0] / Compensation1[0] = 1000000
- p_Premium for Product2 / Object1[0] / Event1[0] / Compensation1[1] = 1000000
- p_Premium for Product2 / Object1[0] / Event1[1] / Compensation1[0] = 1000000
- p_Premium for Product2 / Object1[0] / Event1[1] / Compensation1[1] = 1000000
- p_Premium for Product2 / Object1[1] / Event1[0] / Compensation1[0] = 1000000
- p_Premium for Product2 / Object1[1] / Event1[0] / Compensation1[1] = 1000000
- p_Premium for Product2 / Object1[1] / Event1[1] / Compensation1[0] = 1000000
- p_Premium for Product2 / Object1[1] / Event1[1] / Compensation1[1] = 1000000
- p_Premium for Product2 / Object1[0] / Event1[0] / Compensation2[0] = 100000000

- p_Premium for Product2 / Object1[0] / Event1[0] / Compensation2[1] = 100000000
- p_Premium for Product2 / Object1[0] / Event1[0] / Compensation2[2] = 100000000
- p_Premium for Product2 / Object1[0] / Event1[1] / Compensation2[0] = 100000000
- p_Premium for Product2 / Object1[0] / Event1[1] / Compensation2[1] = 100000000
- p_Premium for Product2 / Object1[0] / Event1[1] / Compensation2[2] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[0] / Compensation2[0] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[0] / Compensation2[1] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[0] / Compensation2[2] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[1] / Compensation2[0] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[1] / Compensation2[1] = 100000000
- p_Premium for Product2 / Object1[1] / Event1[1] / Compensation2[2] = 100000000

68.30. Close the **Test** window.

69. Add Compensations to the layout (D)

69.1. Create Product2PE1 subnode Product2PE1C1

69.1. Right-click on **Product2PE1**. A pop-up dialog appears.
69.2. Select **Insert subnode**. A subnode for Product2PE1 appears.
69.3. Change **Title** to **Product2PE1C1**.
69.4. Set **Multiple** to **Yes**.
69.5. Set **Count Attribute** to **a_Compensation1s**.
69.6. Set **Caption New** to **new compensation1**.
69.7. Set **Caption Delete** to **delete compensation1**.

69.2. Add elements to page Product2PE1C1

69.8. Add element **Label** with name **Compensation1Name** with **Title** as **Compensation1 Name**.
69.9. Add element **Entry field** with name **CompensationName** with **Source** as **a_NameCompensation1**.
69.10. For element **Entry field**: Set **Compute** to **On change**.
69.11. Add element **Label** with name **wwwee** with **Title** as **Product2 Person E1 Compensation1 premium**.
69.12. Add element **Result field** with name **fddfdfd** with **Source** as **p_PremiumP2O1E1C1Each()**.

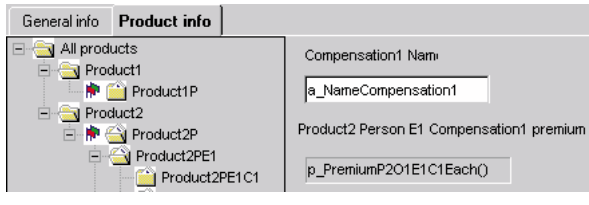


Figure 69.1. Elements for page Product2PE1C1

69.3. Set node Product2PE1C1 identification

69.13. Set **Identification** to **CompensationName**.

69.4. Create Product2PE1 subnode Product2PE1C2

69.14. Right-click on **Product2PE1**. A pop-up dialog appears.
69.15. Select **Insert subnode**. A subnode for Product2PE1 appears.
69.16. Change **Title** to **Product2PE1C2**.
69.17. Set **Multiple** to **Yes**.
69.18. Set **Count Attribute** to **a_Compensation2s**.
69.19. Set **Caption New** to **new compensation2**.
69.20. Set **Caption Delete** to **delete compensation2**.

69.5. Add elements to page Product2PE1C2

69.21. Add element **Label** with name **ddddeees** with **Title** as **Compensation2 Name**.
69.22. Add element **Entry field** with name **Compensation2Name** with **Source** as **a_NameCompensation2**.
69.23. For element **Entry field**: Set **Compute** to **On change**.
69.24. Add element **Label** with name **fdfsd33ee** with **Title** as **Product2 Person E1 Compensation2 premium**.

69.25. Add element **Result field** with name **fdfdfd** with **Source** as **p_PremiumP2O1E1C2Each()**.

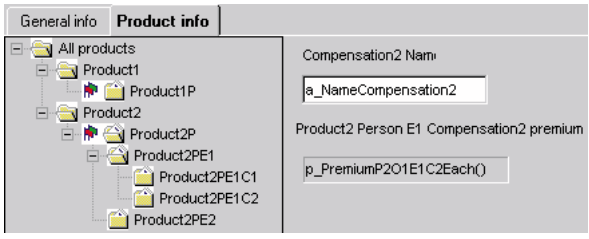


Figure 69.2. Elements for page Product2PE1C2

69.6. Set node Product2PE1C2 identification

69.26. Set **Identification** to **Compensation2Name**.

69.7. Add whichevent result fields to pages Product2PE1, Product2PE1C1, Product2PE1C2

69.27. On page **Product2PE1**: Add element **Result field** with:

69.27.1. **Name**: **weve**

69.27.2. **Source**: **whichevent()**

69.27.3. **Visible**: **No**

69.27.4. **Multiple**: **Yes**

69.28. On page **Product2PE1C1**: Add element **Result field** with:

69.28.1. **Name**: **wevec1**

69.28.2. **Source**: **whicheventC1()**

69.28.3. **Visible**: **No**

69.28.4. **Multiple**: **Yes**

69.29. On page **Product2PE1C1**: Add element **Result field** with:

69.29.1. **Name**: **wobje1c1x**

69.29.2. **Source**: **whichobjectE1C1()**

69.29.3. **Visible**: **No**

69.29.4. **Multiple**: **Yes**

69.30. On page **Product2PE1C2**: Add element **Result field** with:

69.30.1. **Name**: **wevec2**

69.30.2. **Source**: **whicheventC2()**

69.30.3. **Visible**: **No**

69.30.4. **Multiple**: **Yes**

69.31. Rename pages to **PageC1** and **PageC2**.

69.32. Save the layout (Zyx62.vpl).

69.8. Test (D)

69.33. Click the **Test** button.

69.34. Select the **Product info** page. Note the Compensations in the list.

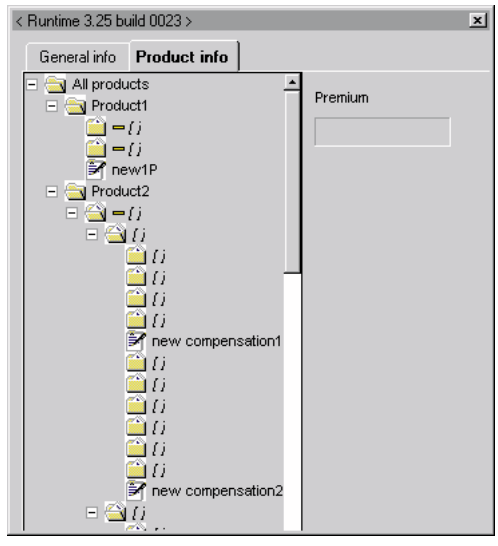


Figure 69.3. Designer test page for the new events

69.35. For all of the objects (persons), events, and compensations: Enter names as shown in the following diagram:

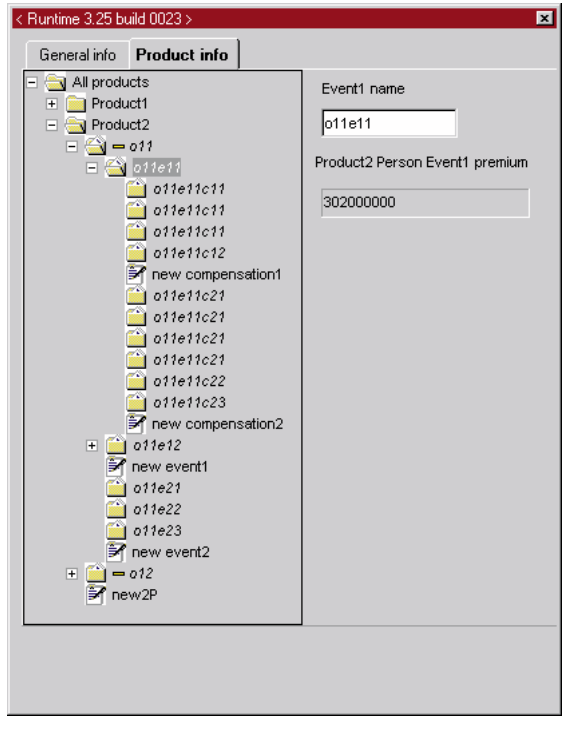


Figure 69.4. Designer test page with names for objects, events and compensations entered

69.36. For **Object1**: Enter **Date of birth** as **1.1.1950**.
69.37. For **Object2**: Enter **Date of birth** as **1.1.1850**. Note that the total premium is **1 208 600 412** (as in

the Workbench test).

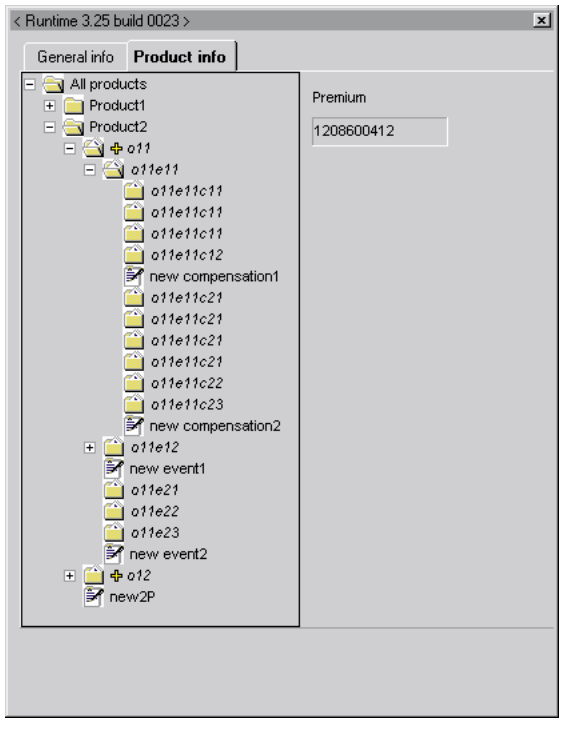


Figure 69.5. Designer test page with required date of birth entered; total premium displayed

70. Add Compensations to Report (RE)

70.1. Add text to report

70.1. Add the following text within and at the end of the bereich Product2 Object x_name2 Event1 x_event:

```
Product2 Object x_name2 Event x_event Compensation1 x_compensation.
Compensation premium = x_compensationpremium.
Product2 Object x_name2 Event x_event Compensation x_compensation.
Compensation premium = x_compensationpremium.
```

70.2. Mark the bold text above as datafields.

70.3. Mark the ZyxHeading1 lines.

70.2. Create new bereichs for added text

70.4. Create 2 new bereichs for the above text as shown in the following diagram:

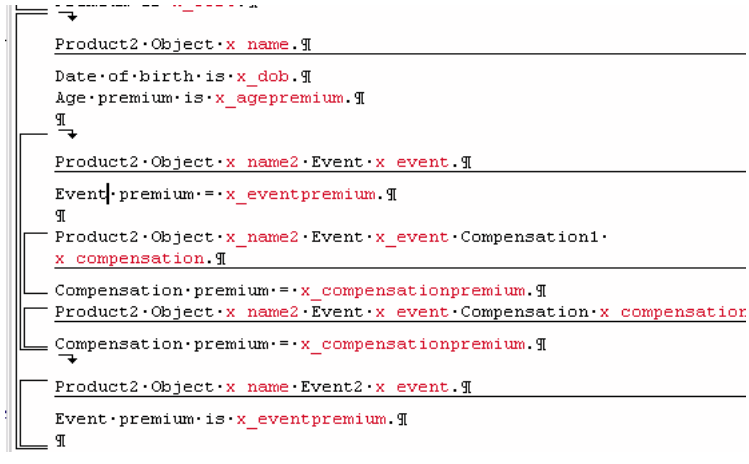


Figure 70.1. New text and new bereichs for compensations in the report

70.5. Enter the following bereichskommando text for the first sub-bereich:

```
Steuerdatei $Datei_PageC1;
if (x_wobj = wobje1c1x) ;
if (x_weve = wevec1) ;
x_name2 = PageProduct2PName;
x_event = EventName;
x_compensation = CompensationName;
x_compensationpremium = fddfdfd;
```

70.6. Enter the following bereichskommando text for the second sub-bereich:

```
Steuerdatei $Datei_PageC2;
if (x_wobj = wobje1c1x) ;
if (x_weve = wevec1) ;
x_name2 = PageProduct2PName;
x_event = Event2Name;
x_compensation = Compensation2Name;
x_compensationpremium = fdfdfd;
```

70.3. Add page breaks

70.7. Add page breaks as shown in the following diagram:

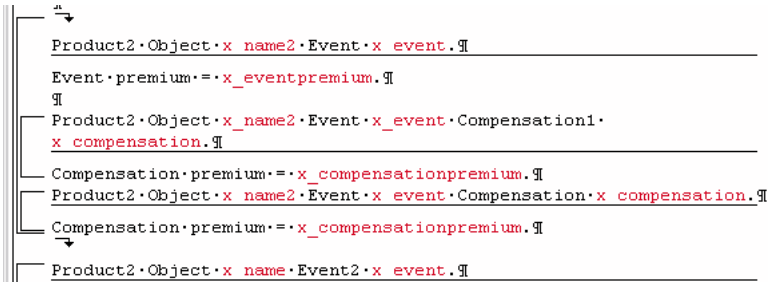


Figure 70.2. New page breaks for the report with compensatinos

70.8. Save the report (Zyx63.cat).

70.4. Test (VF)

70.9. Open the **Zyx consultation** in VFrame.

70.10. Enter the name of the objects and events as shown in the following diagram.

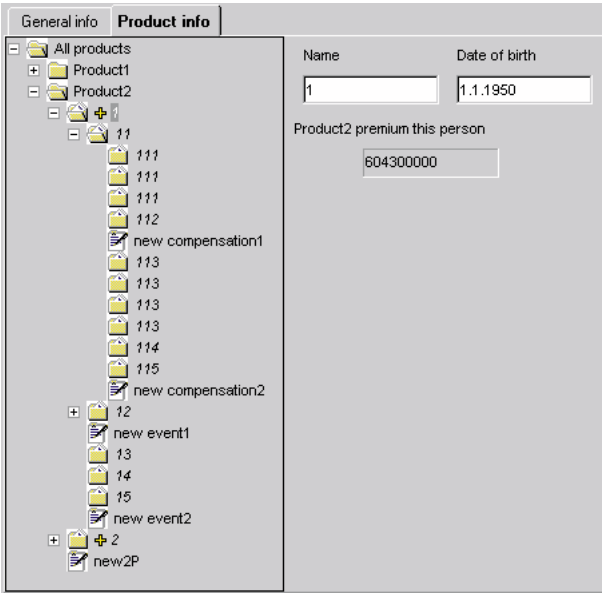


Figure 70.3. Consultation with entered object (person), event and compensation names

70.11. For **1**: Enter **Date of birth** as **1.1.1950**.

70.12. For **2**: Enter **Date of birth** as **1.1.1850**. Note that the total premium of **1 208 600 412** is now displayed.

70.13. Print a report. Note the new pages for Product2 / Person1 / Event 1 / Compensations:

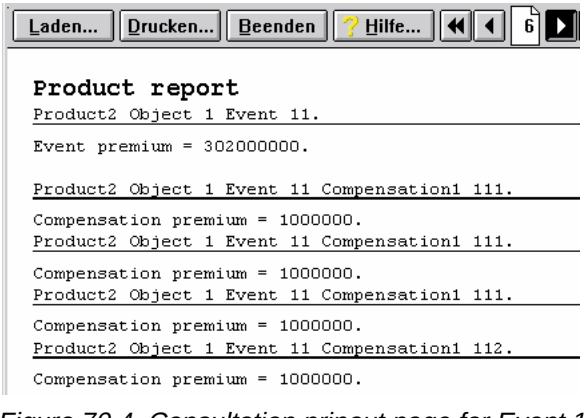


Figure 70.4. Consultation prinout page for Event 1 Consultation 1

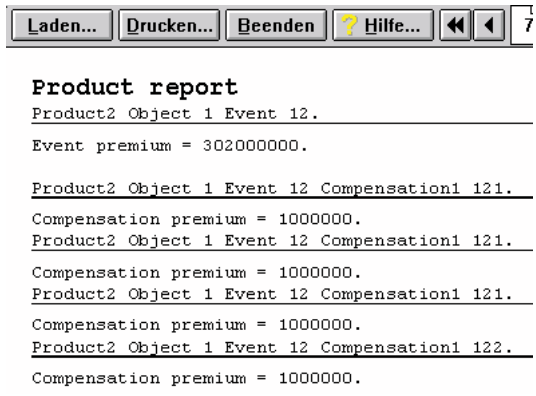


Figure 70.5. Consultation prinout page for Event 1 Consultation 2

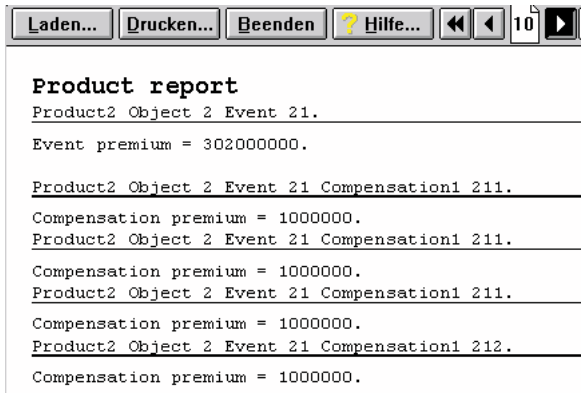


Figure 70.6. Consultation prinout page for Event 2 Consultation 1

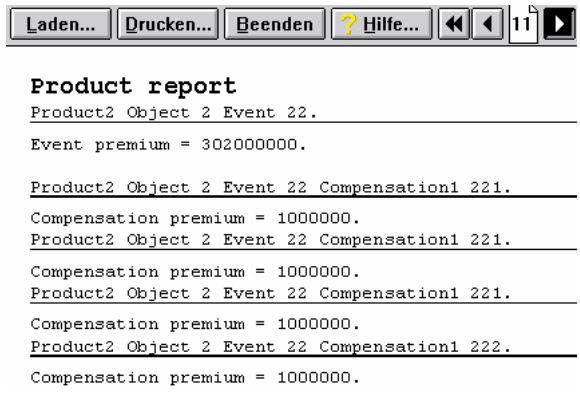


Figure 70.7. Consultation prinout page for Event 2 Consultation 2

71. Components ??

71.1. Create component Object2

71.1. Create Object **Object2**.

71.2. Drag and drop **Object2** from the **Objects** window to the **Components** window. Object2 can now be added to product trees as a component by dragging from the Components window.

Figure 71.1. Component Object2 in the Components window

71.2. Add component Object2 to Product1, Product2

71.3. Drag and drop **component Object2** to **Product1** (note: drop component Object2 on Product1 subnode Object1 so that component Object1 is positioned after subnode Object1).

71.4. Drag and drop **component Object2** to **Product2** (note: drop component Object2 on Product2 subnode Object1 so that component Object1 is positioned after subnode Object1).

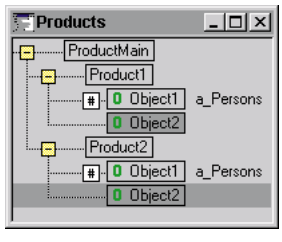


Figure 71.2. Component Object2 in the Product1, Product2 trees

71.3. Specifying inclusion, properties for component Object2 in Products or Components windows

71.5. Double-click on **component Object2** for **Product1** in the **Products** window.

71.6. Set **IncType** to **multiple**. Note that the inclusion type is set in both the Products window and the Components window:

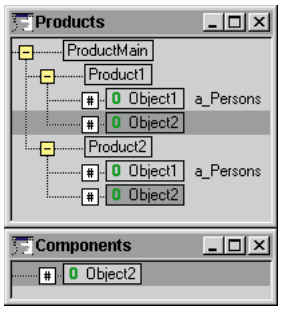


Figure 71.3. Changes to component Object2 in Products window

71.7. Double-click on **component Object2** in the **Components** window.

71.8. Set **IncRule** to **a_Persons**. Note that the inclusion rule is set in both the Products window and the Components window:

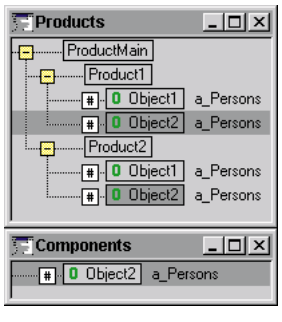


Figure 71.4. Changes to component Object2 in Components window

71.9. Add property **p_Premium** with value **100**.

71.10. Save the model (Zyx64.pms).

71.11. Create runtime model.

71.4. Test (W)

71.12. Click **Test**.

71.13. Click **Compute**. The result of **1412** is displayed. This result includes the following:

- p_Premium for Product1 / Object1[0] = 4 = 1 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- p_Premium for Product1 / Object1[1] = 8 = 2 * 2 * 2 = f_AgePremium * a_Weeks * f_CoverageLevelPremium
- p_Premium for Product1 / component Object2[0] = 100
- p_Premium for Product1 / component Object2[1] = 100
- p_Premium for Product2 / Object1[0] = 0
- p_Premium for Product2 / Object1[1] = 1000
- p_Premium for Product2 / component Object2[0] = 100
- p_Premium for Product2 / component Object2[1] = 100

71.14. Close the test.

71.5. Add Event1 from Events window to Components window

71.15. Drag&Drop **Event1** from the **Events window** to the **Components window**. Note the following:

- Event1 remained in the Events window.
- Event1 was copied as a subnode for the component Object1.

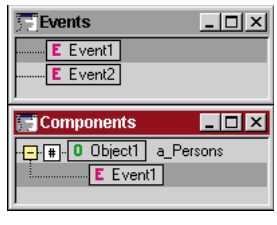


Figure 71.5. Event1 copied from Events window to Components window

71.6. Copy component Object1 to Product2

71.16. Drag&Drop **component Object1** to **Product2**. Note that the subcomponent Event1 was also copied.

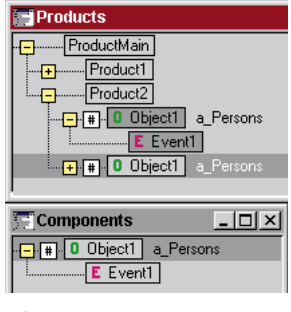


Figure 71.6. Component Object1 copied to Product2

71.7. Copy subcomponent Event1 to Product2 non-component Object1

71.17. Drag&Drop **subcomponent Event1** to the **non-component Object1** in **Product2**.

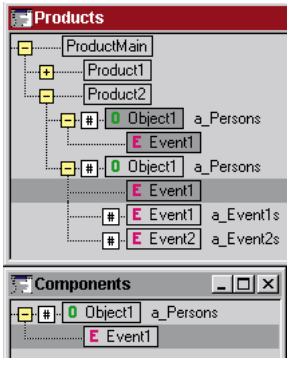


Figure 71.7. Subcomponent Object1 copied to Product2 / non-component Object1

71.8. Promote subcomponent Event1 to stand-alone component

71.18. Right-click on **subcomponent Event1**. A pop-up window appears.

71.19. Select **Promote**. Note that **component Event1** is no longer a subcomponent in either the Products or Components windows.

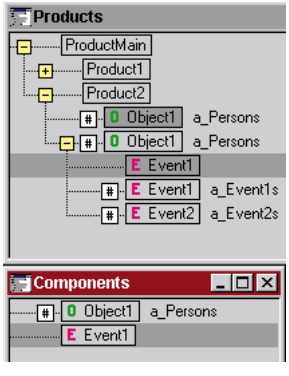


Figure 71.8. Promoted stand-alone component Event1

71.9. Remove component link

71.20. Right-click on **component Event1** in the **Products window**. A pop-up window appears.

71.21. Select **Remove link**. Note that **Event1** is no longer a component in the Products window.

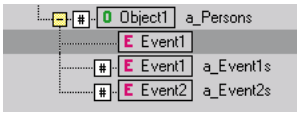


Figure 71.9. Component linked removed for Event1 in the Products window

71.10. Load the previous model version

71.22. From the main menu: Select **Model / Version**. The message **Save modified product model?** appears.

71.23. Click **No**. The **Versions** dialog appears.

71.24. Double-click on the last version.

71.11. Drag & drop Compensation1 to the Components window

71.25. Drag & drop Compensation1 to the Components window. Note that Compensation1 is added as a

subnode of Object2.

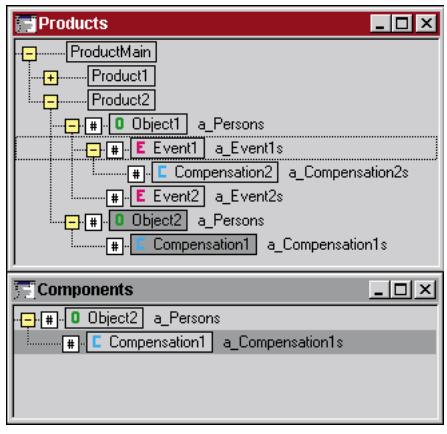


Figure 71.10. Compensation1 dragged&dropped to the components window

71.12. Drag & drop component Compensation1 to the Products window

71.26. Drag & drop component Compensation1 onto **Product2 / Object1 / Event1** in the Products window.

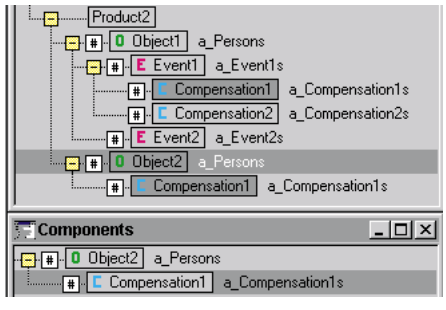


Figure 71.11. Component Compensation1 dragged&dropped onto Product2 / Object1 / Event1

71.13. Promote component Compensation1 in the Components window

71.27. Right-click on component **Compensation1** in the **Components** window.

71.28. Select **Promote**. Note that Compensation1 is no longer a subcomponent of component Object2.

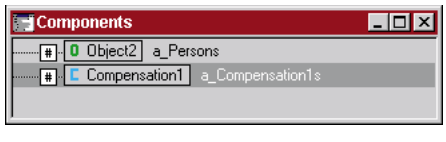


Figure 71.12. Component Compensation1 promoted

71.14. Drag & drop Event1 to the Components window

71.29. Drag & drop Event1 to the Components window. Note that Event1 can only be added as a subcomponent of Object2. Also note that the subnodes (including the Compensation1 component) are also

moved.

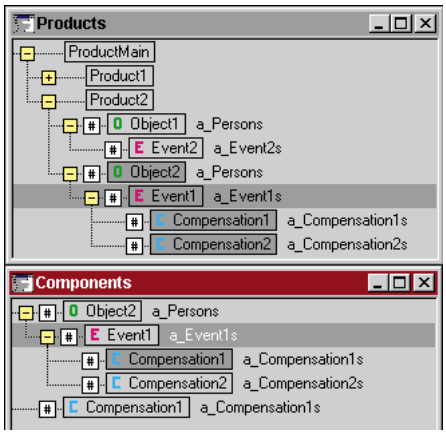


Figure 71.13. Event1 dragged&dropped to the components window

71.15. Drag & drop component Event1 to the Products window

71.30. Drag & drop component Event1 onto **Product2 / Object1** in the Products window.

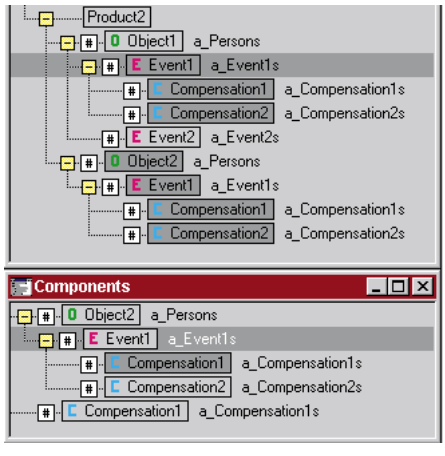


Figure 71.14. Component Event1 dragged&dropped onto Product2 / Object1

71.16. Promote component Event1 in the Components window

71.31. Right-click on component **Event1** in the **Components** window.

71.32. Select **Promote**. Note that Event1 is no longer a subcomponent of component Object2.

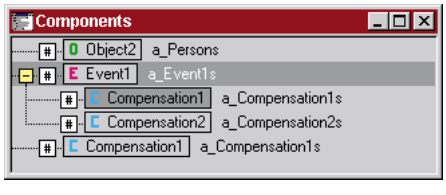


Figure 71.15. Component Event1 promoted

71.17. Test (W, VF)

71.33. Save the model.

71.34. Create a runtime model.

71.35. Test in Workbench and VFrame. Note that the functionality of the model has not changed.

71.18. Remove link to component Event1

71.36. Right-click on component **Event1** in the **Products** window.

71.37. Select **Remove link**. Note that Product2 / Object1 / Event1 is no longer a component in the Pro-

ducts window.

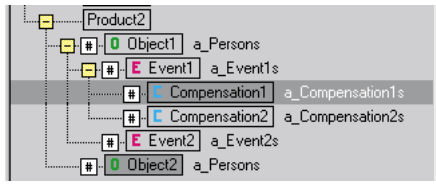


Figure 71.16. Link removed for Product2 / Object1 / Event1

71.19. Remove link to component Compensation1

71.38. Right-click on component **Compensation1** in the **Products** window.

71.39. Select **Remove link**. Note that Product2 / Object1 / Event1 / Compensation1 is no longer a component in the Products window.

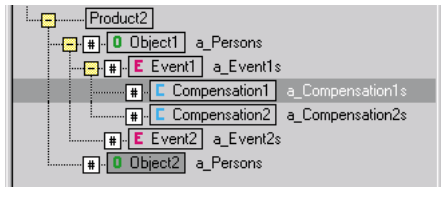


Figure 71.17. Link removed for Product2 / Object1 / Event1 / Compensation1

71.20. Test (W, VF)

71.40. Save the model (Zyx64.pms).

71.41. Create a runtime model.

71.42. Test in Workbench and VFrame. Note that the functionality of the model has not changed.

72. Conclusion

This is the end of the VP/MS tutorial. Having completed this tutorial, you have a good basic understanding of the following:

- How the VP/MS tools (VFrame, Workbench, Designer, Report Editor, Report Viewer) work together
- How to perform basic tasks with VP/MS tools

This tutorial focused on the basics of using VP/MS tools (not on realistic applications). It is now recommended that you continue your VP/MS training with a tutorial in one of the following manuals:

- VFrame User's Guide
- Workbench User's Guide
- Designer User's Guide
- IFOS (Report Editor, Report Viewer) User's Guide

The tutorials in the above manuals contain more advanced examples for typical VP/MS applications. And the User's Guides present detailed information about each tool.

Appendix A

Tools, Directories, Files, File Types

This appendix provides an overview of the following:

- The tools that make up VPMS.
- The directories that contain VPMS files.
- VPMS files.
- File types used by VPMS.

Tools

Workbench (W)

The Workbench is used to create a model file (.pms) and a runtime model file (.vpm). The Test tool is also called from within the Workbench.

Designer (D)

The Designer is used to create a layout file (.vpl). The layout file can be tested from within Designer; however, a runtime model (.vpm; created within Workbench) is required for computations (the Designer cannot perform computations).

VFrame (VF)

VFrame is used for consultations. The consultation brings everything together: The runtime model, the layout, and the report. The VFrame consultation configuration is performed by changing the required configuration information in the C:\VPMS\VFrame\menu*.ini files.

IFOS Report Editor (RE)

RE is used to create a report file (.cat). This file determines how the information entered within VFrame will be printed.

IFOS Report Viewer (RV)

RV is used to create a view (using the format information in the report file).

Directories

Note: This description assumes that the VP/MS components were installed in the recommended default directories.

C: \ Vpms

C: \ VPL_Apps

VPMS application subdirectories. When a VPMS application is installed on your computer, a subdirectory is created in this directory for that application.

C: \ VPL_Apps \ ZYX

Zyx application files, including .pms, .vpc, .vpd, .vpl, .vpm files for the Zyx application.

C: \ VPL_Apps \ ZYX \ data

Zyx application data files (if any).

C: \ VPL_Apps \ ZYX \ help

Zyx application windows help files (if any).

C: \ VPL_Apps \ ZYX \ images

Zyx application image files (if any).

C: \ VPL_Apps \ ZYX \ print.

Zyx application Report Editor .cat files for printing.

C: \ VPL_Apps \ ZYX \ struct

Zyx application .ini files for .dll calls (if any).

C: \ VPL_Apps \ ZYX \ vorgang

Zyx application vorgang files.

C: \ Vpms \ Designer

All Designer files (including Designer Test).

C: \ Vpms \ Vframe

Vframe executive and help.

C: \ Vpms \ Vframe \ menu

ini files for the Vframe menus (including consultation).

C: \ Vpms \ wbench

All Workbench (including Workbench Test) and IFOS files.

Files

Cafrg.exe

C: \ VPMS \ wbench

Report viewer.

consult.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Consultation.

Ifosre.exe

C: \ VPMS \ wbench

Report editor.

menubar.ini

C: \ VPMS \ Vframe \ menu

Definition of Vframe main menu.

menufens.ini

C: \ VPMS \ Vframe \ menu (??)

Definition of submenu items for VFrame main menu item Fenster.

menuhelp.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Help.

menuinfo.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Infodesk.

menukund.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Customer.

Vds.exe

C: \ VPMS \ Designer

Designer.

vds_tx32.exe

C: \ VPMS \ Designer

Designer test.

verwalt.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Administration.

vframe32.exe

C: \ VPMS \ Vframe

VFrame.

vorgang.ini

C: \ VPMS \ Vframe \ menu

Definition of submenu items for VFrame main menu item Process.

Vpms32.exe

C: \ VPMS \ wbench

Workbench.

vpmsdl32.dll

C: \ VPMS \ Wbench

Creates a runtime from the .pms file.

vpmsdll.dll

C: \ Winnt \ System32

16-bit interface to the 16-bit IFOS.

vpmsin32.dll

C: \ Winnt \ System32

??.

vpmsinfo.dll

C: \ Winnt \ System32

Provides access to the attributes in the designer.

Vpmste32.exe

C: \ VPMS \ wbench

Workbench test.

xplconf.ini

C: \ VPMS \ Vframe \ menu

Definitions for VFrame.

ZYX.cat

C: \ VPL_Apps \ ZYX \ print

ZYX report format.

ZYX.hlp

C: \ VPL_Apps \ ZYX \ help

Help for the ZYX consultation in VFrame.

ZYX.pms

C: \ VPL_Apps \ ZYX

Zyx model. Klartext datei.

ZYX.vpc

C: \ VPL_Apps \ ZYX

The compiled (runtime) .vpl layout file.

ZYX.vpd

C: \ VPL_Apps \ ZYX

Mapping of ZYX layout components and ZYX model attributes.

ZYX.vpl

C: \ VPL_Apps \ ZYX

Zyx layout.

ZYX.vpm

C: \ VPL_Apps \ ZYX

Zyx runtime model.

File types

.cat

Report file.

.pms

Model (includes all versions of the model).

.vpd

Mapping of layout components and model attributes.

.vpc

Compiled (runtime) .vpl file.

.vpl

Layout.

.vpm

Runtime model.

Appendix B

Glossary

This glossary defines terms that are presented throughout this manual.

Product tree

The structure (internal) of the black box.

Versicherungs Andwendung Architektur

Objekt, Ereignis, Leistung

Active tree

Different types of cover are found side by side in the product tree. By defining inclusion rules, the product modeler defines which parts of the tree will be included in the calculations under certain conditions and which not. The part of the product tree where the inclusion rules are fulfilled in a specific situation is known as the active tree. If a property is used several times within the active tree, only the property in the active tree will be included in the evaluation.

Attribute

The rules and calculations featuring in this product relate to data which describe e.g. the characteristics of an insured object (e.g. kW of a ve-hicle). This data is mapped as attributes in the VP/MS. In other words, all input fields on the application program's user interface are defined as attributes. Checks and default values can be defined for each attribute.

Automatic aggregation

If several partial premiums are calculated for a product, these partial premiums must be aggregated into a total premium. This aggregation process is performed automatically by complying with the following convention: A superordinate level of the product tree states that the total premium is the same as the total of the partial premiums of the superordinate levels. Avoid manual aggregation of premiums in the form partial premium 1 + partial premium 2 + partial premium 3.

Auxiliary element

Where differences occur for the first time in the product tree, the product modeler can integrate freely selectable auxiliary elements in the product tree. Example: Two types of cover are represented by auxiliary elements.

Basic catalog for Objects, Events, Compensation

The objects, events and compensation which can be used in products are each defined in a basic catalog (Objects window, Events window, Com-pensation window on the Workbench). This is the only place where they can be changed. They are linked dynamically with the products and product components.

Compensation

Standard element of the product tree which specifies what compensation is given and how much. Examples: Daily allowance, pension, restoration costs, disposal costs, indemnification, refund of costs.

Components

In order to avoid having to map identical product parts several times, product components can be created a single time and reused at different locations in the product tree. Changes to product components apply throughout the product model.

Example: Dog owner's liability in household and building insurance, refund of costs of electrical appliances for types of household insurance cover, identical refund of costs for different health insurance rates.

Database

This is a structured collection of data. It enables the gathering and query-ing of information (e.g. master customer data) using specific criteria. The query is made using special query languages such as SQL (Structured Query Language) or ODBC. It is possible to address ODBC-compatible programs (e.g. Access, Excel) in VP/MS as standard.

Editor

The tool which allows the product modeler to enter and change rules and calculations. It is launched by double-clicking a line of the Inspector. Automatic assignment of different colors within the Editor enhances the legibility of formulae (e.g. functions are depicted in blue, attribute or table names in green, values in dark red and text in blue-green).

Event

Standard element of the product tree which specifies what an object is to be insured against. Examples: Accident, start of pension, theft, glass breakage, storm, compensation claim, vehicle damage. The frequently used

term ’risk’ is not used, since e.g.’start of pension’ cannot be termed a ’risk’.

Functions

Like Excel, VP/MS has a number of predefined "standard functions", e.g. if (if; then; else), error (error message), days (date).

Properties which are used several times can also be defined as functions (in the Functions window) by the product modeler. Like the standard functions, these "user-defined functions" can be used in every property. The product modeler can greatly reduce his workload and facilitate the update of product definitions by defining functions for recurring calcula-tions and rules.

It is also possible to use external programs (e.g. actuarial programs in the code of another programming language) as well as user-defined functions.

Inclusion rule

Specifies the condition under which an element of the product tree is part of the active tree. An inclusion rule must always be stated in an element – except where elements are of inclusion type 'mandatory'. In the simplest of cases, the inclusion rule is specified by an attribute.

Inclusion type

For each element in the product tree, the conditions under which it forms part of the active tree are stipulated. The following inclusion types are possible: mandatory, optional, mutually exclusive and multiple. The condi-tions under which an optional element can be included are defined in an inclusion rule.

An element forms part of the active tree if the inclusion rules are satisfied for all superordinate elements and the element itself is either mandatory or satisfies its inclusion rule.

Inclusion value

It is possible to define different product parts where only one part can be included. These product parts are identified by an inclusion rule, e.g. specification of the TypeofCover attribute. The inclusion value specifies the type of cover for which this element is included.

Inspector

Tool for the detailed view of all elements of the product model. The pro-duct modeler can inspect all entries in detail by double-clicking all ele-ments of the model or by selecting the elements and pressing the Enter key.

Object

Standard element of the product tree which defines which person or object is to be insured. In order to identify the object to be insured, ask yourself the following question: Who or what will the event affect? Examples: Person, building, car, animal.

ODBC

Open Database Application Programming Interface. Software interface regulating the exchange of data between an application program and databases. VP/MS can access ODBC-compatible databases such as Access via an ODBC interface.

Platform

This term is used to designate different computer systems in terms of their system architecture. These differ as regards their CPU (central processing unit), hardware components, operating system and available software version. Examples of different operating systems are: Win 3.11, 95, NT, OS2, host systems, Unix.

PMS

Filename extension of the insurance product models created with Work-bench.

Product

Insurance products can be represented in the form of a tree which con-tains the elements product, objects, events and compensation (in this order). This tree-like representation of an insurance product can be crea-ted by the product modeler in the Products window, the "heart" of the VP/MS Workbench. Classes of insurance always represent products from the perspective of VP/MS. The product tree maps every aspect of the cover information (both mandatory and optional product elements) and does so in a way which is easy to understand for all target groups. All calculation regulations and rules can be defined here.

Product tree

The objects which can be insured in a product, the events relating to these which can be insured, and the associated compensation are represented in the form of a tree in the VP/MS Workbench. The product tree is a com-

plete representation of the cover provided by a product.

Property

The rules and calculations relating to a product are known as properties in VP/MS. Properties can be defined as formulae for an element in the product tree or as formulae for attributes (=data fields relevant for the product rate). Properties consist of a property name and a calculation formula for determining the value of this property. Examples of properties include: checks, defaults, premiums.

Reference model

In section "Standardizing Attributes, Functions, Tables", the subordinate product model is known as the reference model. Attributes, functions and tables relating to this reference model are available to the top model (superordinate model).

Runtime module

To run the product model on different target systems, a system-independent file is created with the filename extension "VPM". This VPM file is also known as a "runtime" module and uses a platform-specific DLL file which enables it to run on all platforms!!

This DLL, also known as a runtime, corresponds to a virtual machine which runs the runtime modules (VPM) on different target systems. This runtime is available for the following operating systems, for example: Windows 3.x (16-bit runtime = vpmsdll.dll), Windows 95 (32-bit runtime = vpmsdl32.dll) and OS2.

DLL is an abbreviation for Dynamic Link Library. These are object libraries which can be loaded dynamically (i.e. as required). Several programs can access a loaded DLL simultaneously.

Submodel

In section "Creating Bundled Products", the linked product model is known as the submodel. The product tree, attributes, functions and tables of the submodel (of the linked model) are available to the top model (superordinate model) by means of a link.

Table

Within tables, you can map mortality tables, premiums, descriptions (long text), error messages and default values in selection fields. These default values (e.g. different types of cover in the Product selection field) are also known as selection options or selection list entries. All elements in the product tree can access a table.

VP/MS can be used to access external dBase tables, internal VP/MS tables and external ODBC-compatible programs (Access, Excel).

Top model

The superordinate product model is known as the top model in sections "Standardizing Attributes, Functions, Tables" and "Creating Bundled Products".

VPM

Filename extension for runtime modules created from the product models. The filename extension when generating runtime modules is VPM.

VP/MS

VP/MS means Visual Product / Modeling System and is a tool for insuran-ce product modeling. VP/MS contains the Workbench, runtime module, test and mass test. VP/MS is an expert system which works in the back-ground of application programs. The application programs (=client appli-cations, e.g. field service programs, broker software, host applications) access the expert system.

Workbench

The product developer is equipped with a graphical workstation complete with intuitive graphical interface for modeling insurance products. This workstation is known as the Workbench in VP/MS. The product developer uses the Workbench to edit and manage products (define new products, change existing products). The insurance product models created with the Workbench are given the filename extension PMS.

List Of Figures

Index