

VP/MS Designer User's Guide

Table of contents

Table of contents	2
Contact	11
Copyright and trademarks	12
About this User's Guide	13
Other sources of information	14
Release notes	15
Release notes v3.30.....	15
Release notes v3.25.....	16
Installation	18
Designer minimum system requirements.....	18
System limitations	18
Installation step-by-step.....	18
Installing Designer (Designer not already installed).....	19
Installing Designer (Designer already installed).....	19
Tutorial.....	21
Basics	23
T01. Example layouts and models	23
T02. Specify extension defaults	24
T03. Create new layout	25
T04. Add element to layout	26
T05. Test layout.....	27
Workbench model	28
T06. Assign a model to the layout.....	28
T07. Open the model in the Workbench	28
Calculation	29
T08. Add model attribute to layout	29
T09. Add model property to layout	30
T10. Test result calculation	31
Workareas, pages, header/footers	32
T11. Add 2nd workarea to layout	32
T12. Add application header/footer	33
T13. Add 2nd page to workarea	34
T14. Add workarea header/footer.....	35
Editing with Designer.....	37
T15. Find	37
T16. Cut / Copy / Paste	38
T17. Groups.....	38
T18. Autosize.....	41
Multiple layouts/models	42
T19. Multiple layouts	42
T20. Multiple models	44
External DLL	45
T21. Calling a DLL from the layout.....	45
Documenting a layout.....	47
T22. Commenting layout components	47

T23. Generating layout report	48
T24. Creating layout page screenshot	50
T25. Compare layouts	51
Tooltips	52
T26. Tooltips.....	52
Datatypes.....	53
T27. Datatype: Single-line text	53
T28. Datatype: Multiline text.....	54
T29. Datatype: Boolean.....	55
T30. Datatype: Number	55
T31. Datatype: Currency	56
T32. Datatype: Date	57
T33. Datatype: Mask	58
T34. Datatypes: Arrays: Tables.....	58
T35. Datatypes: Arrays: Iteration.....	59
T36. Datatypes: Arrays: Multiple pages	61
Entering/selecting data.....	63
T37. Conversion list.....	63
T38. Dynamic Combobox.....	65
T39. Element order.....	66
T40. Start attribute.....	67
T41. Parameter search.....	68
T42. Data indicator	69
Recomputing result	70
T43. Calculate: Page change	70
T44. Calculate: Focus change.....	71
T45. Calculate: Data entry.....	72
T46. Calculate: Pushbutton click.....	73
Visibility	74
T47. Visibility: Design-time	74
T48. Visibility: Run-time static	75
T49. Visibility: Run-time dynamic	75
Availability	77
T50. Availability: Static	77
T51. Availability: Dynamic	78
Where to go from here	79
Tools and dialogs.....	81
Main dialog.....	81
Title bar.....	81
Menu bar	82
Toolbar: Tools (visual palette).....	92
Toolbar: Palette (common).....	94
Toolbar: Alignment (groups).....	95
Toolbar: Header/footer tools.....	97
Main dialog: Design area.....	97
Main dialog: Status bar.....	97
Dialog: Options.....	98
Dialog: Options / Application	98
Dialog: Options / Workarea	100
Dialog: Options / Page	101
Dialog: Options / Element.....	103

Dialog: Options / Useability	104
Dialog: Options / Designer settings	105
Dialog: Options / IFOS	106
Inspector	106
Test dialog.....	107
Dialog: Find / Find next	107
Dialog: Elements (re)ordering.....	107
Dialog: Autosize-Tool	108
Dialog: Autosize / Proportional Factor Automatic.....	108
Dialog: Autosize / Proportional Factor Manual.....	109
Dialog: Autosize / font Map	109
Autosize fontBrowser	109
Autosize fontList Save/Load.....	109
Dialog: Synchronisierungs-Tool.....	109
Dialog: Synchronisations-Report	110
Dialog: Data elements	111
CRF preparation (print) dialog	112
Report type	112
Sorting	112
Range.....	112
Output On.....	113
Printer Settings	113
Select page for screen shot dialog	113
Concepts.....	114
VP/MS components.....	114
Model.....	114
Runtime model	115
Layout.....	115
Compiled layout.....	115
Consultation.....	115
Report.....	116
Layout components	116
Application	117
Application Header/Footer.....	117
Workarea Header/Footer.....	118
Page	119
Element	120
Concept: EditField	121
Concept: ComboBox	121
Concept: CheckBox.....	122
Concept: Radio button.....	122
Concept: Push Button	122
Concept: Label	122
Concept: Border	123
Concept: Grid	123
Concept: ExtendedGrid	126
Concept: 3d Border	129
Concept: Line	130
Concept: Multimedia element.....	130
Concept: ResultFields	130
Concept: RadioGroup.....	132

Concept: Graphics element.....	133
Add / delete layout components	143
Application header/footer add/delete	144
Workarea insert / shift / remove	144
Workarea header/footer add/delete	147
Page insert / shift / remove.....	148
Element add / delete	151
ElementGroup add / delete	152
Properties.....	155
Layout and component format.....	156
External connections	156
Layout runtime functionality.....	157
Data connections model/layout.....	157
Model attributes	157
Model attribute properties.....	158
Model properties.....	160
Data types.....	160
Text (single-line).....	160
Multi-line Text	160
Boolean	161
Number.....	161
Currency.....	162
Date	162
Mask.....	162
Entering / selecting data.....	163
Entering data	164
Selecting data.....	164
Concept: Element order	165
Start attribute	165
Parameter search.....	166
Data indicator	168
Displaying data.....	169
Displaying data: EditField.....	169
ResultField.....	170
Dynamic Label / Border.....	171
Grid / ExtendedGrid.....	172
Dynamic Graphic.....	172
Compute result property.....	172
Change page.....	172
Pushbutton	173
Enter data	174
Change focus	174
Re-compute attribute default/label	175
Enter data.....	176
Change focus	177
Visibility	178
Design-time visibility (element levels)	178
Runtime layout visibility	178
Static visibility	178
Dynamic visibility	179
Availability	182

Results list	182
Static Availability	185
Dynamic availability	185
Arrays.....	186
Table.....	186
Iteration (with Grid)	191
Multiple pages	193
Subdialogs	202
Subdialog: Open.....	203
Subdialog: Close	203
Media element.....	205
DLL call	205
Layout help / tooltips	206
Layout help	206
Tooltips.....	206
Layout comment / print / screen shot.....	209
Comments	209
Layout print (CRF Preparation)	210
Report output: Screen	217
Report output: Printer	217
Layout screenshot	217
Layout test.....	217
Common tasks	218
Task: Versioning	218
Task: Searching	218
Task: Aligning.....	218
Examples	219
Example: Dynamic drop-down list (combobox).....	219
Example: Radiogroup.....	219
Example: Combobox.....	220
Example: Conversion list.....	220
Example: Data indicator	221
Example: Visualize	221
Example: Dynamic	222
Example: Visible.....	222
Example: Multiple	223
Example: Page autostart	223
Example: Secondary dialogs	224
Example: Compute results	224
Example: Expand/collapse page tree	225
Example: Available.....	225
Example: Strict check.....	226
Example: Grid	226
Example: Wizard	227
Properties	229
Properties: Component list	229
Properties: Application	229
Properties: Workarea	230
Properties: Page.....	230

Properties: EditField	231
Properties: ComboBox	232
Properties: CheckBox.....	232
Properties: RadioButton	233
Properties: PushButton	234
Properties: Label	234
Properties: Border	235
Properties: Grid	235
Properties: ExtendedGrid.....	236
Properties: 3D Border.....	237
Properties: Line	237
Properties: Multimedia element.....	237
Properties: ResultField	238
Properties: RadioGroup.....	238
Properties: Graphics element.....	239
Properties: All.....	240
Property: Action	242
Property: ActivColor	243
Property: Alignment.....	243
Property: AppActiveColor	244
Property: Attribute	245
Property: AVAILABLE-Property.....	246
Property: Background.....	247
Property: Background color(R,G,B).....	248
Property: Bounds.....	249
Property: Calculation	250
Property: Caption Delete	251
Property: Caption New	252
Property: ChkDynamic	252
Property: ChkOtherAvailability.....	253
Property: ChkOtherVisibility	254
Property: ChkOwnAvailability	254
Property: ChkValidation.....	255
Property: Color	256
Property: Column.....	256
Property: Columns moveable	257
Property: Compute	258
Property: ComputeResult	258
Property: Conversion List.....	259
Property: ConvertAttr.....	260
Property: Count attribute	261
Property: Currency symbol.....	262
Property: Data column.....	263
Property: Data elements.....	264
Property: DataIndicator	265
Property: Decimal position	265
Property: Default font	266
Property: Defolding.....	267
Property: Devisualize	268
Property: DI Rule.....	269
Property: Dialog-Description	270
Property: DII-Description	270

Property: ElemActiveColor	271
Property: Element3D	271
Property: Font.....	272
Property: Footer.....	274
Property: Format (element)	274
Property: Format (application).....	275
Property: Frame font	276
Property: Frame width	278
Property: FrameMoveable.....	278
Property: Graphics type.....	279
Property: Group	280
Property: Header	281
Property: Header font	282
Property: Height (application).....	283
Property: Height (Header/Footer).....	284
Property: HelpID	285
Property: Identification.....	286
Property: Identification Grid.....	287
Property: Indexed	287
Property: Iteration count	289
Property: Iteration property.....	289
Property: Iteration type	290
Property: Key column	291
Property: Left	292
Property: Legend placement	292
Property: Level	293
Property: List (...;...)	294
Property: Mask	295
Property: MaxFrameWidth	296
Property: Maximum	297
Property: Media file	298
Property: MinFrameWidth	298
Property: Minimum	299
Property: Multiline.....	300
Property: Multiple.....	301
Property: Name	301
Property: Name (Workarea/Page).....	302
Property: New Session.....	302
Property: Next line	304
Property: Number of columns.....	305
Property: Off conversion List.....	306
Property: Off symbol.....	306
Property: On symbol.....	307
Property: Order.....	309
Property: Position (Footer)	309
Property: Position (Header).....	310
Property: Productmodel.....	310
Property: Property	311
Property: Property for	312
Property: Series.....	313
Property: Source.....	313
Property: StartAttr.....	314

Property: Strict check	315
Property: Style (application)	315
Property: Style (workarea).....	316
Property: Tab font.....	316
Property: Tab color	317
Property: Title	318
Property: Tooltip (static/dynamic).....	318
Property: Tooltip (static only).....	319
Property: Tooltip text	320
Property: Top.....	321
Property: Type (entry elements).....	321
Property: Type (result elements).....	323
Property: Type (visual elements).....	324
Property: Type (GraphicsElement).....	325
Property: Type (Grid).....	325
Property: Type (ExtendedGrid)	325
Property: Type (Grid column)	326
Property: Type (ExtendedGrid column).....	327
Property: Usage.....	328
Property: Value.....	328
Property: Version.....	329
Property: Version info.....	330
Property: Visible (element)	331
Property: Visible (Workarea/Page).....	331
Property: Visualize	332
Property: Width (application)	332
Property: Width (Grid/ExtendedGrid)	333
Property: X-axe name	334
Property: Y-axe name	334
Trouble-shooting.....	336
Window not visible.....	336
Element not visible	336
Model attribute not available	336
Header/footer (application/workarea) not visible.....	337
Header/footer missing	337
FAQ (frequently-asked questions)	339
Overlapping elements	339
Attributes from more than 1 model.....	340
Glossary.....	341
Directories and files	345
Directories	345
c:\vpms	345
c:\vpms\designer	345
c:\vpms\vframe	345
c:\vpms\vframe\menu	346
c:\vpms\wbench.....	346
c:\vpl_apps	346
c:\vpl_apps\zyx.....	346
c:\vpl_apps\zyx\data	346
c:\vpl_apps\zyx\help	346
c:\vpl_apps\zyx\images	346

c:\vpl_apps\zyx\print	346
c:\vpl_apps\zyx\struct	346
c:\vpl_apps\zyx\vorgang	347
Files	347
cafrg.exe	347
consult.ini	347
empty.vpm	348
ifosre.exe	348
menubar.ini	348
menuburo.ini	348
menufens.ini	349
menuhelp.ini	349
menuinfo.ini	349
menukund.ini	349
vds.exe	350
vds_tx32.exe	350
verwalt.ini	350
vframe32.exe	350
vorgang.ini	350
vpms32.exe	351
vpmsdl32.dll	351
vpmsdll.dll	351
vpmsin32.dll	351
vpmsinfo.dll	352
vpmste32.exe	352
xplconf.ini	352
File types	352
.avi	353
.bmp	353
.cat	353
.hlp	354
.jpg	354
.pms	354
.tif (tiff)	354
.vpc	355
.vpd	355
.vpg	355
.vpl	356
.vpm	362
.wav	362

Contact

Mynd SoftwareConsult GmbH
Taubenholzweg 1
D-51105 Köln
Germany
Tel : (+49) (0) 221 - 8029 - 0
FAX : (+49) (0) 221 - 8029 - 999
Email: info@mynd.de
Web: www.mynd.de

Copyright and trademarks

Copyright

Copyright 2000 PMS MICADO. All rights reserved.
VP/MS Designer User's Guide. 19 June 2000.

Trademarks

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

About this User's Guide

Text conventions

The Designer is always used with the Workbench. Workbench attributes can sometime have string values that include double quotations marks (") (the quotation mark is not a string delimiter, but actually part of the string). This is very important, because leaving out the quotation marks in the Workbench value will generate no errors. Therefore, this help often delimits string values with less-than (<) and greater-than (>) signs.

Other sources of information

Help

Online help is available for:

- Designer (also on the CD ROM as \docs\english\designer\designer.chm (HTML compiled format)).
- Workbench (also on the CD ROM as \docs\english\workench\workbench.hlp (Windows 95 help format)).
- IFOS Report Editor (also on the CD ROM as \docs\english\ifos\reporteditor.hlp (Windows 95 help format)).

Designer help is also available in non-compiled HTML help format (for viewing in a browser) on the CD ROM as \docs\english\designer\designer.htm.

Manuals

The following manuals are available on the CD ROM:

- \docs\english\workbench\designer.pdf: Designer User's Guide (this document).
- \docs\english\vpms\vpms_getting_started.pdf: VP/MS Getting Started manual. Includes a step-by-step tutorial for quickly gaining experience with VPMS.
- \docs\english\workbench\workbench.doc: Workench User's Guide.

Release notes

- Version 3.30
- Version 3.25

See also:

- Designer version information in the Status bar
- Designer version information in the Info dialog

Release notes v3.30

HTML Help

Help for Designer is now in Microsoft HTML Help format.

Application/Workarea header/footer

Layout components now include application header/footer and workarea header/footer. For details see Adding/Deleting layout components subsections:

- Application header/footer
- Workarea header/footer.
- Header/footer toolbar.

Workarea/Page visibility with property <Visible>

Workareas and pages visibility can be controlled using property Visible (previously visibility of workareas and pages could only be controlled using Visualize/Devisualize).

Any row can be deleted in ExtendedGrid

The ExtendedGrid now supports inserting or deleting of a row at any location in the ExtendedGrid (previously only the last row could be inserted or deleted).

Saving page views to .bmp file ("export screen shot")

Layout pages can now be saved to .bmp files by selecting menu option File / Export screen shot to open the Select page for screen shot dialog. Concept details.

Option dialog available from menu item "Window / Options..."

If no layout is open in the Designer: The Options dialog can be opened by selecting from the main menu "Window / Options...".

Extended layout report functionality

The functionality of the layout reports has been extended.

The CRF preparation dialog and the IFOS options dialog have been updated accordingly.

License support

Software license functionality has been added.

The license is implemented using registry entry

HKEY_LOCAL_MACHINE/SOFTWARE/CAFGmbH/vpms Designer/VPMS/. If the required information is not entered in fields LicenceName and LicenceNR, then only demo-version functionality of Designer will be available.

License support requires service.dll.

Note: Select menu item Help/Info for information about what version (full or demo) of Designer is being used.

Release notes v3.25

New runtime model initialization file: vds_rt.ini

vds_rt.ini is new.

Section MAIN

DYNAMIC_LIST_BUFFER_SIZE (default = 255). Default buffer size for dynamics.

Section PERFORMANCE

INITIALIZE_ATTRIBUTES (default = ON).

- ON: Non-initialized attributes are initialized before saving data (XPLGetData).
- OFF: Non-initialized attributes are NOT initialized before saving data (XPLGetData).

RECALCULATE_VISIBLE_PAGES_ONLY (default = OFF).

- ON: Invisible pages are reinitialized and recalculated when saving data (XPLGetData).
- OFF: Invisible pages are NOT reinitialized and recalculated when saving data (XPLGetData).

LOADDATA_COUNTATTR_RECALCDYNAMIC (default = ON).

When creating a new page for a multiple node, the checking of dynamics can now be disabled. Checking the dynamics can significantly reduce performance for large dynamic lists.

- ON: All dynamics are recalculated.
- OFF: Dynamics are NOT recalculated.

PRELOAD_ACTION_ELEM_ONLY (default OFF).

The model attributes for elements with properties Visualize/Devisualize (CheckBox's and RadioButtons) can be optionally initialized at startup. Optional initialization means that the model attribute is only initialized if a page or workarea has been specified in the corresponding element property's Visualize or Devisualize list.

- ON: Attributes are initialized only for CheckBox's and RadioButtons with page/workarea selected in the Visualize or Devisualize list.
- OFF: Attributes are initialized for all CheckBox's and RadioButtons.

New warning messages

- "Missing identification grid." Graphics identification grid has not been defined.
- "Automatically generated name has not been changed." Automatically generated name for page or element was not redefined.
- "No elements are defined on page."
- "No identification element is defined for multiple page." Property Identification for multiple page is not defined.

Installation

- System requirements
- System limitations
- Step-by-step

Designer minimum system requirements

Design-time:

- IBM PC 486
- Windows 95 or Microsoft Windows NT 4
- 32MB RAM
- 20MB available hard disk space

Run-time:

Windows 3.1

- 486/50 Mhz
- 8 MB memory

Windows 95

- Pentium/90 Mhz
- 20 MB memory

Windows/NT

- Pentium/133 Mhz
- 32 MB memory

System limitations

Technical limitations (16-bit version):

- Maximum name length: 60 Characters
- Maximum number of results per page: 54
- Maximum number of EditFields per page: 200

Installation step-by-step

The installation procedure differs slightly, depending on whether Designer has already been installed.

- Installing from CD ROM (Designer not already installed)
- Installing from CD ROM (Designer already installed)

Note: It is always recommended to install VPMS components in the following order:

- VFrame
- Workbench
- Designer

Installing Designer (Designer not already installed)

1. Double click on E:\Designer\Setup.exe ("E" = CD ROM drive letter). The <Welcome> dialog appears.
2. Click Next. The <Get Registration Information> dialog appears.
3. In the <Licensename> field: Enter your name.
4. In the <Licensnumber> field: Enter your license number. Note: If no license is entered, a trial version will be installed.
5. Click Next. The <Registration Information> dialog appears.
6. 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.
7. Click Next. The Backup replaced files? dialog appears. Yes is selected.
8. 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.
9. Click Next. The files for IFOS are copied to your harddisk. The dialog Installation Completed! appears.
10. Click Finish.

Designer is now installed on your computer and available from:

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

Installing Designer (Designer already installed)

1. Double click on E:\Designer\Setup.exe ("E" = CD ROM drive letter). The <Welcome> dialog appears.
2. Click Next. The <Get Registration Information> dialog appears.
3. In the <Licensename> field: Enter your name.
4. In the <Licensnumber> field: Enter your license number. Note: If no license is entered, a trial version will be installed.
5. Click Next. The <Registration Information> dialog appears.
6. 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.
7. Click Next. The Backup replaced files? dialog appears. Yes is selected.
8. Click Next. The files for Designer are copied to your harddisk.
9. Click Finish.

Designer is now installed on your computer and available from:

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



Tutorial

This section presents a tutorial that introduces the basic functionality of the Designer.

All example layouts and models in this tutorial are available on the CD-ROM in directory /docs/english/designer/tutorial.

Workbench models are required for working with Designer. However, if you have no working knowledge of the Workbench, you can still complete this tutorial with the supplied pre-compiled Workbench models.

For a complete step-by-step tutorial of the complete VP/MS toolset (including Designer, Workbench, VFrame and the Report Editor), consult the VP/MS Getting Started manual.

Overview

Basics

- T01. Example models and layouts.
- T02. Specify extension defaults.
- T03. Create a layout
- T04. Add an element (label) to the layout.
- T05. Test the layout (using the Designer Test application).

Workbench model

- T06. Assign a model to the layout.
- T07. Open the model in the Workbench.

Model attributes/properties in the layout

- T08. Add model attributes to the layout.
- T09. Add a model property to the layout.
- T10. Test the layout.

Workareas, pages, header/footers

- T11. Add a second workarea to the layout.
- T12. Add application header/footer to the layout.
- T13. Add a second page to a workarea.
- T14. Add workarea header/footer to the layout.

Edit tools

- T15. Find
- T16. Cut / Copy / Paste
- T17. Groups
- T18. Autosize

Multiple layouts/models

- T19. Multiple layouts
- T20. Multiple models

External DLL

- T21. Calling a DLL from the layout

Layout documentation

- T22. Component comments
- T23. Layout report
- T24. Page screenshot
- T25. Layout comparison (synchronize)

Tooltips

- T26. Tooltips

Data types

- T27. Single-line text
- T28. Multi-line text
- T29. Boolean
- T30. Number
- T31. Currency
- T32. Date
- T33. Mask

- T34. Table arrays
- T35. Iteration arrays
- T36. Multiple-page arrays

Data entry

- T37. Conversion list
- T38. Dynamic combobox
- T39. Element order
- T40. Start attribute
- T41. Parameter search
- T42. Data indicator

Result calculation

- T43. Page change
- T44. Focus change
- T45. Data entry
- T46. Pushbutton click

Component visibility

- T47. Design-time
- T48. Run-time static
- T49. Run-time dynamic

Component availability

- T50. Static
- T51. Dynamic

Where to go from here

Basics

The first part of this tutorial introduces information to help you get the most from the tutorial. You will also create and test your first simple layout.

T01. Example layouts and models

The completed layout and any required files for the layout (such as the model) are provided in directory docs/english/designer/tutorial. The name of the completed layout and required files match the page number at the top of the help page. For example, completed layout for T08 is 08.vpl and the completed model is 08.pms.

It is recommended to backup the \tutorial directory and then work through the tutorial, overwriting any files that you edit or create.

Example

Note: If you are totally new to Designer and are planning to do each section starting from the beginning, then go now to the next page now (the text below includes terminology and concepts that you would not require for completing the tutorial).

Assume that you want to start the tutorial at page T08 and that the following files exist:

1. 08.vpl: Completed design-time layout for page T08.
2. 08.vpc: Completed runtime layout for page T08.
3. 08.pms: Completed design-time model for page T08.
4. 08.vpm: Run-time model for T08.
5. 07.vpl: Completed design-time layout for page T07.
6. 06.pms: Completed design-time model for page T06 (assume that changes to the model were not required on page T07 (and thus 07.pms does not exist)).

You would have the following options:

- Simply open the T08 runtime layout 08.vpc in Designer Test.
- Open the T08 design-time layout 08.vpl and then click the test button to regenerate and open the T08 runtime layout 08.vpc in Designer Test.
- Open the T07 design-time layout 07.vpl and follow the step-by-step instructions in T08.
- (Workbench experience required) Add the required model changes described in T08 to design-time model 06.pms, save the model as 08.pms, compile the model as 08.vpm, and then test the layout with the model (instructions for using the Workbench is outside the scope of this Designer tutorial).

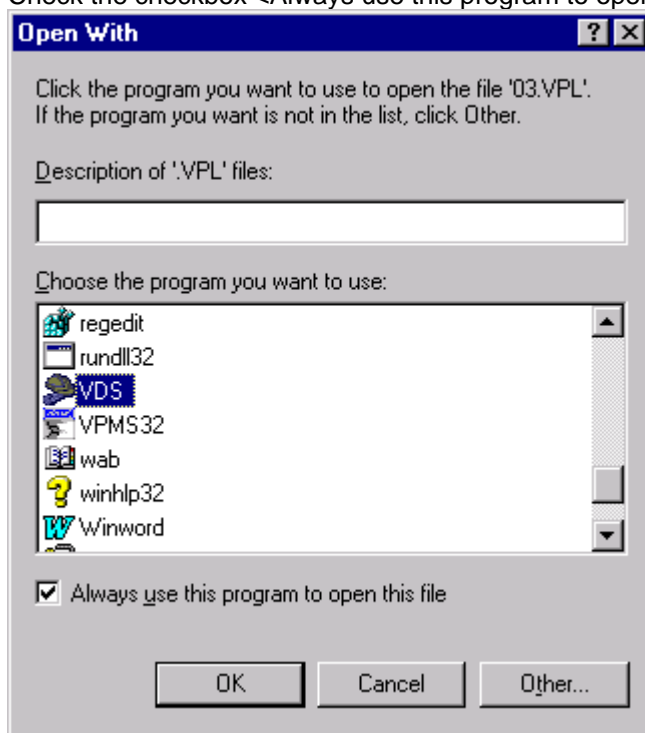
T02. Specify extension defaults

You will now specify the default applications for extension defaults for Designer and (if installed) Workbench. This will allow you to open, for example, a .vpl file in the Designer by simply double-clicking on the .vpl file. This will make completion of the tutorial much faster.

Design-time layout (.vpl)

Do the following:

1. Open directory \docs\english\designer\tutorial.
2. Press and hold the Shift key.
3. Right click on any .vpl file. A pop-up menu appears.
4. Select "Open With...". The "Open with" dialog appears.
5. In the list <Choose the program you want to use>: Select <VDS>. NOTE: VDS should be in the list. If it is not, then click on the <Other...> button and select C:\VPMS\Designer\vds.exe.
6. Check the checkbox <Always use this program to open this file>.



7. Click "OK". The file is opened in the Designer.
8. Close the Designer.

Run-time layout (.vpc)

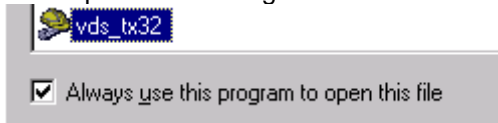
A run-time layout (.vpc) will be created from the design-time layout (.vpl) and displayed in the Designer Test application by clicking on the <Test> icon in the Designer.

However, by assigning extension ".vpc" to application C:\VPMS\Designer\vds_tx32.exe, you can simply double-click on a .vpc file to open the run-time layout in the Designer Test application .

Do the following:

1. Open directory \docs\english\designer\tutorial.
2. Press and hold the Shift key.
3. Right click on any .vpc file. A pop-up menu appears.
4. Select "Open With...". The "Open with" dialog appears. Note: An icon for vds_tx32.exe does not exist in the list.

5. Click <Other...>. The <Open With...> dialog appears.
6. Double-click on C:\Vpms\Designer\vds_tx32.exe. Note that an icon for vds_tx32 appears in the <Open with> dialog:




7. Check the checkbox <Always use this program to open this file>.
8. Click "OK". The file is opened in the Designer Test.
9. Close the Designer Test.

Design-time model (.pms)

Assign design-time models (.pms) to C:\VPMS\WBench\Vpms32.exe. Note: This is only necessary if during the tutorial you want to open the design-time models (not necessary to complete the tutorial).

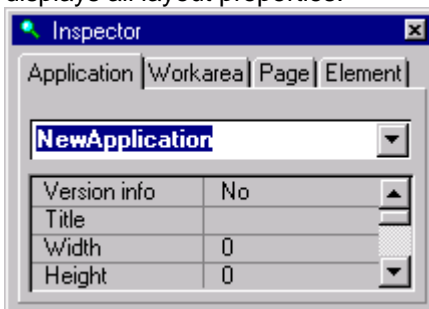
T03. Create new layout

To create a new layout:

1. Select from the Start menu: <Programs / CAF VPMS / VPMS Designer>. The Designer opens.
Note: If the Start menu item is not available: Start vds.exe.
2. Click on the <Create new layout> icon . A new layout is opened. Note: If the layout is not fully visible: From the Designer menu bar: Select <Window / Cascade>.



3. If the Inspector is not visible: From the menu bar: Select <Options / Inspector>. The Inspector displays all layout properties.



4. In Inspector tab Application: Set <Width>=<100>.
5. Set <Height>=<50>. Note the appearance of a dotted red line indicating the boundaries of the layout.

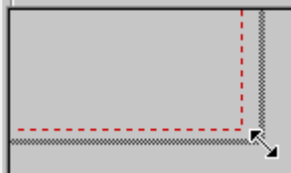


Note: If the line is not visible: From the main menu select <Options / Limit line>.

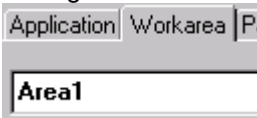
6. Move the cursor to the lower right corner of the box marked by the red dotted line. Note that the cursor becomes an arrow:



7. Press and hold the left mouse button.
8. Move the cursor to resize the layout.

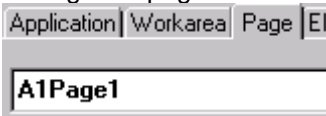


9. Select the Inspector tab <Workarea>.
10. Change the workarea name to "Area1":

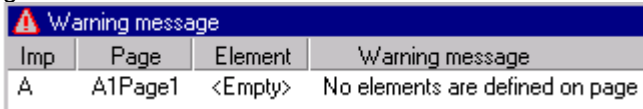


Note: Retype the name even if the name is already <Area1>. Otherwise a warning will be generated that the default name has not been changed.

11. Select the Inspector tab <Page>.
12. Change the page name to <A1Page1>.



13. Select <File / Save as> and save the file as 03.vpl. Note the warning message that is generated:




Note: Saving the file as 03.vpl also creates the following files:

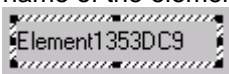
- 03.vpc: Runtime layout.
- 03.vpd: Interface file.
- 03.bak: Layout backup.

T04. Add element to layout

You will now add a Label element to the layout.

Do the following:

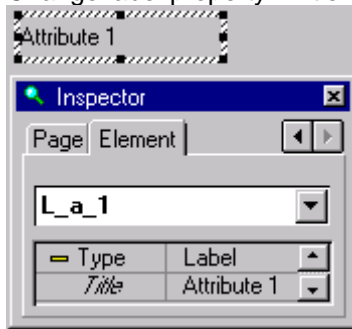
1. If the layout created in T03 is not open in the Designer: Open 03.vpl.
2. Click on the icon <New label> .
3. Click in the layout inside the red dotted line. A Label is added to the layout. Note the default name of the element (the name will be different for your layout).



Whenever an element is added to the layout, the element is assigned a default name. This name should always be changed (if the name is not changed, a warning message will be generated).

4. Change the label name to <L_a_1>.

- Change label property <Title> to <Attribute 1>.



- Click on the Label to select it.
- Position the cursor over any of the solid squares on the Label boundary. Note that the cursor becomes an arrow:




- Press and hold the left mouse button.
- Move the cursor to resize the Label.
- Save the layout as 04.vpl.

T05. Test layout

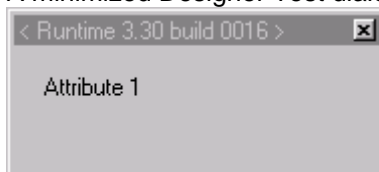
A run-time layout (.vpc) must have a run-time model (.vpm) in order to be tested (with the Designer Test tool). The run-time model normally includes the definitions of model properties and attributes that are referenced in the layout.

Your simple layout doesn't yet reference any model properties or attributes; however, it still requires a run-time model. Our layout doesn't define a required model (this will be demonstrated later in this tutorial). However, by default a layout without a defined run-time model uses the model empty.vpm in directory \vpms\designer (where vds.exe is located). empty.vpm is an empty run-time model included with Designer, and as the name suggests, doesn't define any properties or attributes.

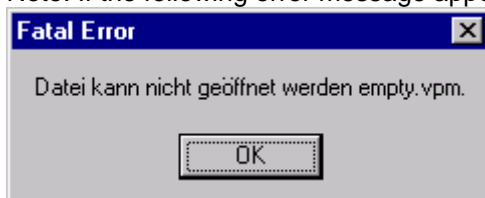
Do the following:

- Open layout 04.vpl if it is not already open in Designer.
- Click on Toolbar icon "Test application" . This automatically generates the .vpc (run-time layout) file.

A minimized Designer Test dialog appears:



Note: If the following error message appears:

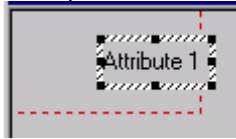


then the default runtime model (empty.vpm) was not found. empty.vpm should have been copied to the required directory during installation. To solve this problem, copy \docs\english\designer\tutorial\empty.vpm to the Designer main directory (\VPMS\Designer).

- Close Designer Test.

Note: If the Label was not fully within the boundaries marked by the red dotted line (limit line), then the layout would appear with scroll bars (the scroll bars would also take up space in the layout). For

example, for the following design-time layout



the resulting runtime layout would be



Workbench model

A layout with any computational functionality (business logic) requires a run-time model. The runtime model defines the business logic. This part of the tutorial introduces the basics for working with models.

T06. Assign a model to the layout

The layout must have a runtime (.vpm) model (if no model is specified, then the default model empty.vpm is used). Whereas empty.vpm must be located in directory \VPMS\Designer, any other model must be located in the same directory as the layout. The model is specified with the application property <Product model>.

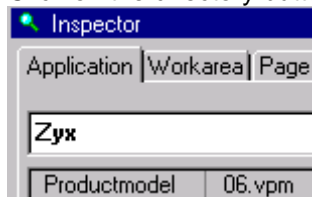
See also: Detailed information about the layout/model connection.

Do the following:

1. If the layout created in T04 is not open in the Designer: Open 04.vpl.
2. In Inspector tab <Layout>: Click in the cell to the right of <Product model>. Note that a directory button appears.



3. Click on the directory button. Select the model /docs/english/designtutorial/06.vpm.



4. Save the layout as 06.vpl.

T07. Open the model in the Workbench

Throughout this tutorial you can use the supplied runtime (.vpm) workbench models. However, in order to select model attributes from a drop-down list within the Designer, the design-time model (.pms) must be the current model open in the Workbench.

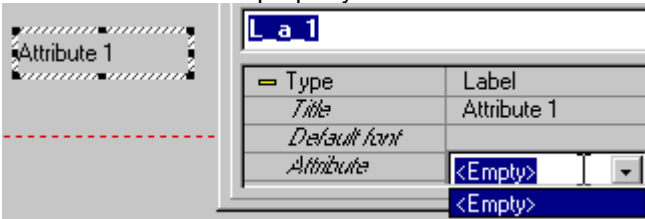
Note: You do not have to select from the drop-down list; you can simply type in the attribute names manually. In this case you do not need to open the design-time model in the workbench.

The following assume that the workbench is set as the default application for opening .pms files (see T02).

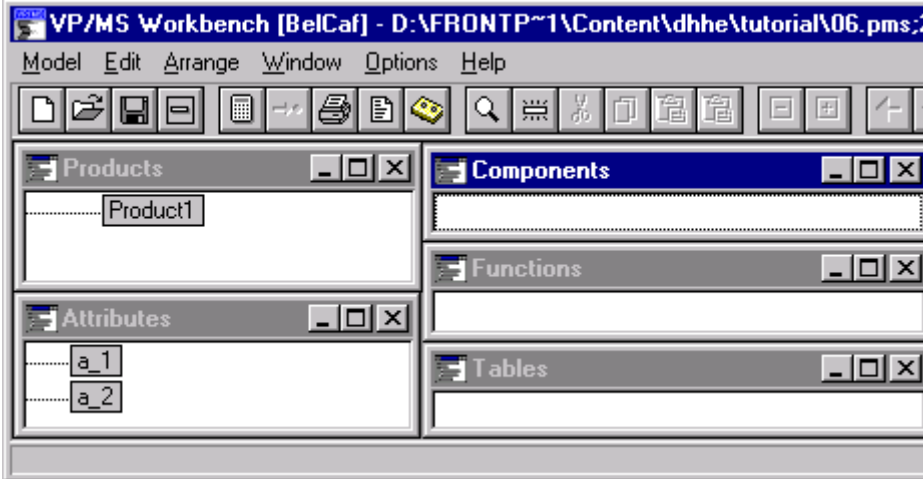
Do the following:

1. If the layout created in T06 is not open in the Designer: Open 06.vpl.
2. Select the Label <Attribute 1>.

- Click on the value for property <Attribute>. Note that no attributes are available.

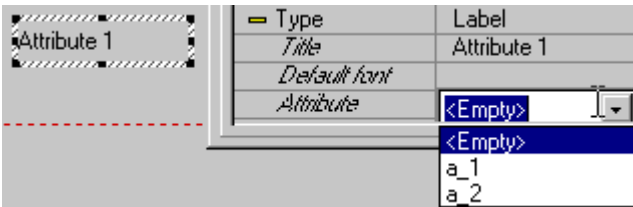


- Double-click on 06.pms. The workbench opens for 06.pms:



Note the 2 attributes <a_1> and <a_2>.

- Click on the value for property <Attribute> in the Designer. Note that still no attributes are available.
- From the Designer menu: Select <Options> / <Workbench synchronisation>.
- Click on the value for property <Attribute> in the Designer. Note that the attributes are now available.




Note: Do not actually select a new value.
Do not close the Workbench.

Calculation

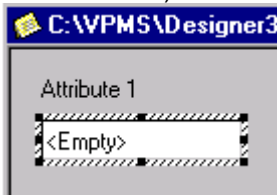
The layout created in the previous sections contained only a label. In this part you will add layout components that will allow you to input data to the model and display result output from the model.

T08. Add model attribute to layout

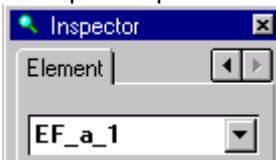
You will now add 2 EditField's to the layout for entering data to the model.
Do the following:

- If the layout created in T06 is not open in the Designer: Open 06.vpl.
- Click on Toolbar icon "EditField" .

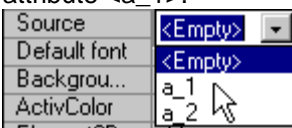
- Click in the layout. An EditField is added to the layout (note that the EditField is the element with the focus).



- In the Inspector: Define a name for the element by entering <EF_a_1> in the drop-down list at the top of Inspector tab "Element" (make sure that the EditField is selected).



- In the Inspector tab "Element": Select from the drop-down list for property <Source> model attribute <a_1>.



Note that the model attribute name is displayed in the EditField:

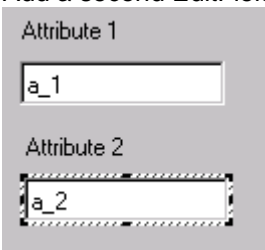


NOTE: If no model attributes are available



then the model is probably not open in the Workbench. Open 06.pms and synchronize with the layout as described in T07.


- Add a second Label with <Name> = <L_a_2> and <Title> = <Attribute 2>.
- Add a second EditField with <Name> = <EF_a_2> and with <Source> = <a_2>.



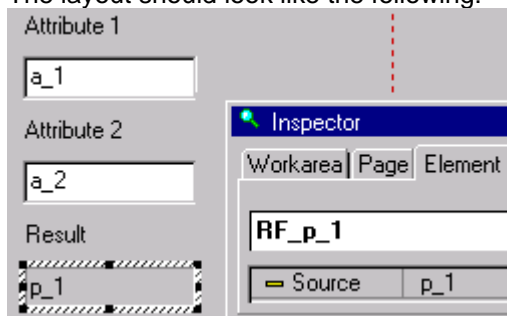
- Save the layout as 08.vpl.

T09. Add model property to layout

You will now add a ResultField to the layout for displaying result data from the model. Do the following:

- If the layout created in T08 is not open in the Designer: Open 08.vpl.
- Add a Label with <Name> = <L_p_1> and <Title> = <Result>.
- Add a ResultField () with <Name> = <RF_p_1> with <Source> = <p_1>. Note: <p_1> will not be available in the <Source> drop-down list, because <p_1> is a model "property". You must type in "p_1".

The layout should look like the following:



4. Save the layout as 09.vpl.

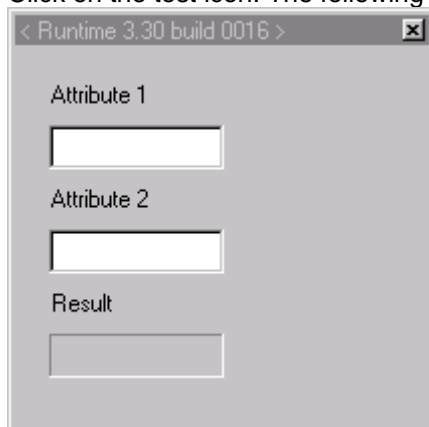
T10. Test result calculation

You will now do your first test of basic calculation with Designer. Data you enter in the EditField's will be input to the runtime model. The result from the runtime model will be displayed in the layout ResultField.

Note: Data input to the model and display of data output from the model can be controlled with layout properties. This will be demonstrated later in this tutorial.

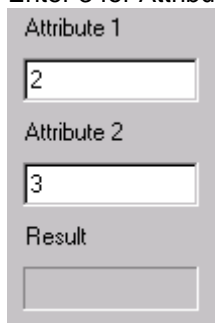
Do the following:

1. If the layout created in T09 is not open in the Designer: Open 09.vpl.
2. Click on the test icon. The following runtime layout appears:



Note: If there are any unsaved changes in the design-time layout, you will be prompted to save the changes.

3. Enter 2 for Attribute 1.
4. Enter 3 for Attribute 2. Note that no result is displayed:



The reasons for this will be explained later in this tutorial.

- Click in the upper EditField to change the focus. Note that the result is now displayed.

Attribute 1	2
Attribute 2	3
Result	6

- Close the test dialog.

Workareas, pages, header/footers

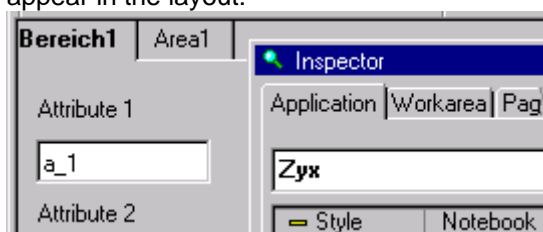
Most layouts contain too much information for a single page. A layout can contain multiple workareas, each of which can contain multiple pages. In addition there can be a header and/or footer for the entire layout and/or for each workarea.

T11. Add 2nd workarea to layout

The current layout consists of a single workarea that contains a single page. In this section you will add a second workarea to the layout.

Do the following:

- If the layout created in T09 is not open in the Designer: Open 09.vpl.
- In the Inspector: In tab <Application>: Set property <Style> = <Notebook>. Note that 2 tabs appear in the layout:



Note: The tab text initial values for your layout will probably be different from those shown above.

- For the first (leftmost) workarea: Set <Name> = <WA1>.
- Set <Title> = <Workarea1>.



- For the second workarea: Set <Name> = <WA2>.
- Set <Title> = <Workarea2>.
- Save the layout as 11.vpl.
- Test the layout.

- Click on the tabs to change the displayed workarea.



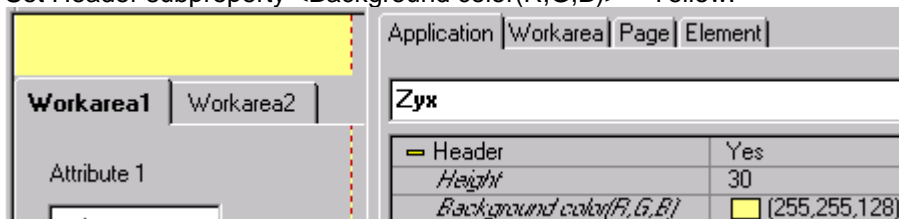
T12. Add application header/footer

A layout can have a header and/or footer that is/are always visible. The header/footer can contain any elements.

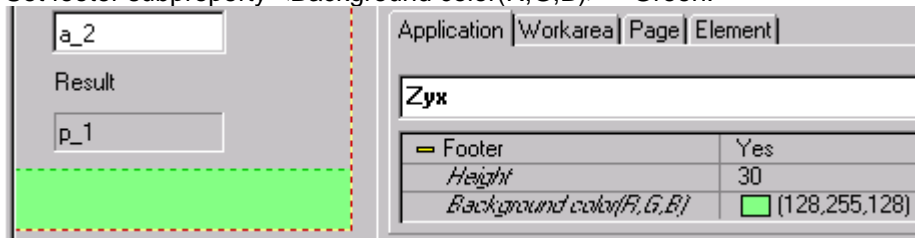
Important: If the header or footer contains no elements and the layout is closed, then the empty header and/or empty footer is deleted from the layout (even if the .vpl file is saved during the close).

You will now add an application header and footer. Do the following:

- If the layout created in T11 is not open in the Designer: Open 11.vpl.
- Set application property <Header> = <Yes>. A header appears in the layout.
- Set Header subproperty <Height> = <30>.
- Set Header subproperty <Background color(R,G,B)> = Yellow.



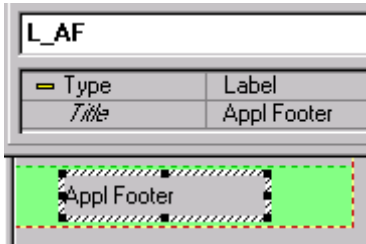
- Set application property <Footer> = <Yes>. A footer appears in the layout.
- Set footer subproperty <Height> = <30>.
- Set footer subproperty <Background color(R,G,B)> = Green.



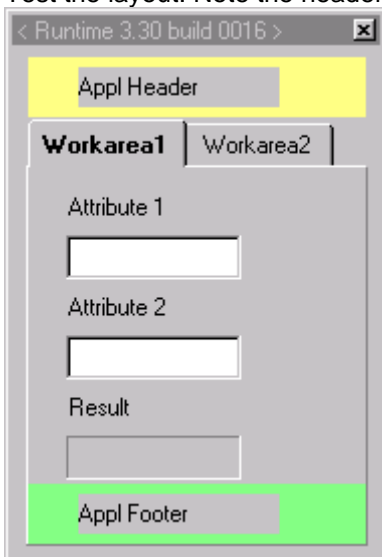
- Add a label to the header with <Name> = <L_AH> and <Title> = <Appl Header>.



9. Add a label to the footer with <Name> = <L_AF> and <Title> = <Appl Footer>.



10. Save the layout as 12.vpl.
11. Test the layout. Note the header and footer:

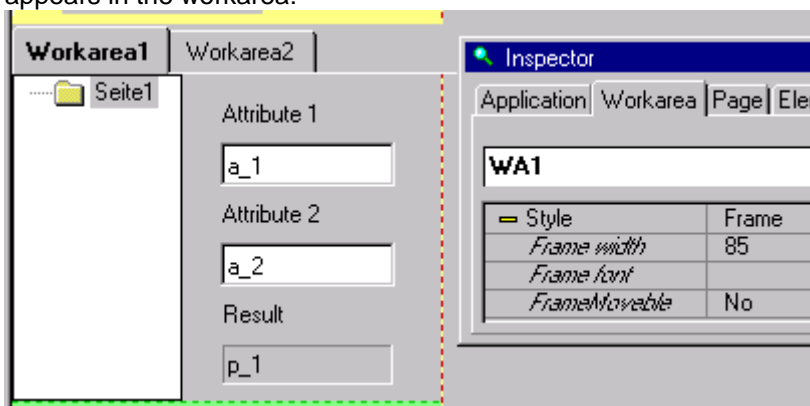


T13. Add 2nd page to workarea

A workarea can contain any number of pages. The pages are accessible by clicking on an icon in the page frame.

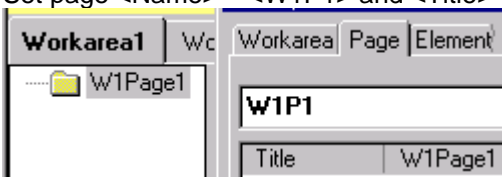
Do the following:

1. If the layout created in T12 is not open in the Designer: Open 12.vpl.
2. Select Workarea1.
3. In the Inspector: In tab Workarea: Select for property Style "Frame". Note that a frame appears in the workarea:

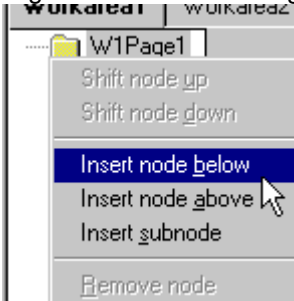


Note: The name of the page ("Seite1" above) may be different in your layout.

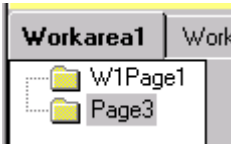
- Set page <Name> = <W1P1> and <Title> = <W1Page1>.



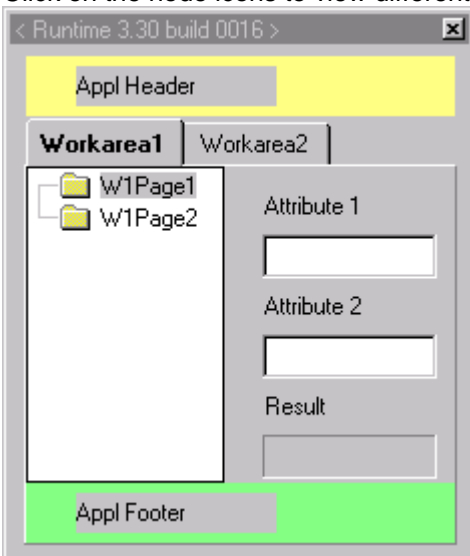
- Right-click on the W1Page1 icon. The following popup menu appears:



- Click on <Insert node below>. A new "node" appears in the frame tree:



- For the second page: Set <Name> = <W1P2> and <Title> = <W1Page2>.
- Save the layout as 13.vpl.
- Test the layout.
- Click on the node icons to view different pages in Workarea1.



T14. Add workarea header/footer

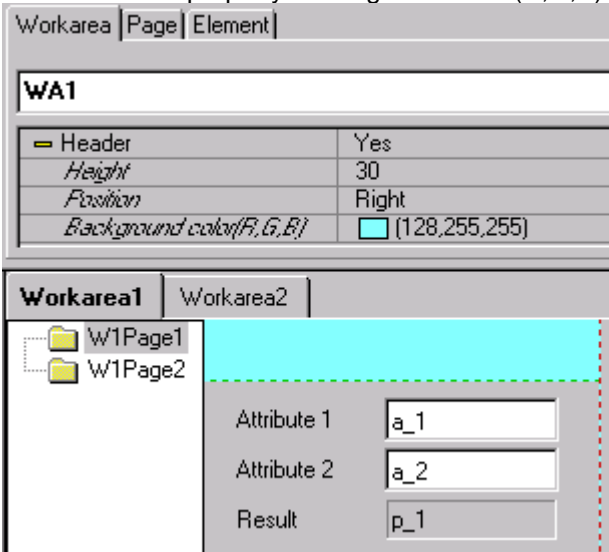
A workarea can have a header and/or footer that is/are visible when the workarea is selected. The header/footer can contain any elements.

Important: If the header or footer contains no elements and the layout is closed, then the empty header and/or empty footer is deleted from the layout (even if the .vpl file is saved during the close).

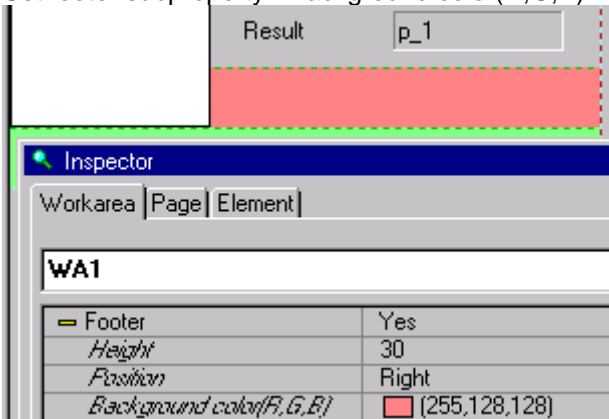
You will now add a workarea header and footer. Do the following:

- If the layout created in T13 is not open in the Designer: Open 13.vpl.
- Select workarea Workarea1.
- Set workarea property <Header> = <Yes>. A header appears in the workarea.
- Set Header subproperty <Height> = <30>.

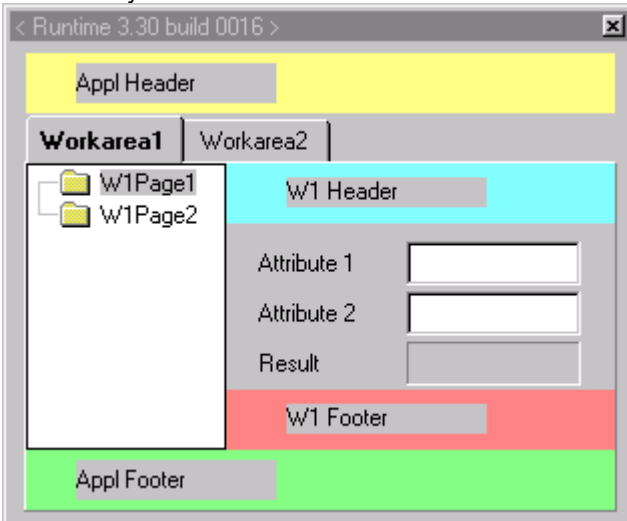
- Set Header subproperty <Background color(R,G,B)> = Light blue.

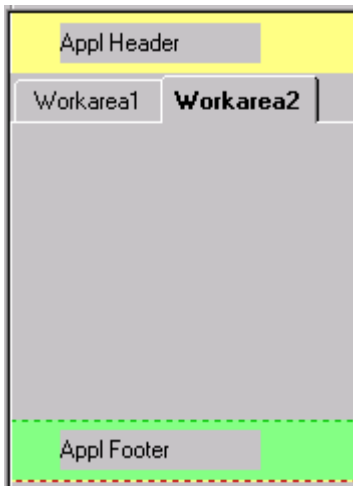


- Set workarea property <Footer> = <Yes>. A footer appears in the workarea.
- Set footer subproperty <Height> = <30>.
- Set footer subproperty <Background color(R,G,B)> = Orange.



- Add a Label to the Header with <Name> = <L_WH> and <Title> = <W1 Header>.
- Add a Label to the Footer with <Name> = <L_WF> and <Title> = <W1 Footer>.
- Save the layout as 14.vpl.
- Test the layout. Note the Workarea1 header and footer.:





Editing with Designer

The Designer provides various tools for working with elements and groups of elements in the layout.

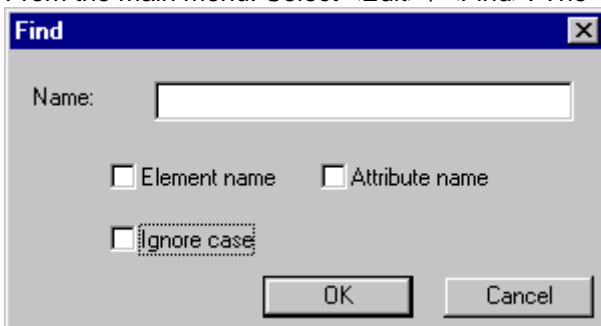
T15. Find

You can search for layout elements:

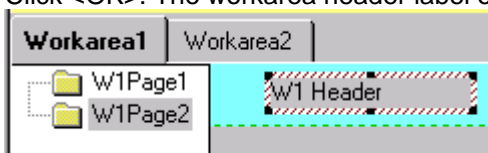
- On a page via the element name
- Anywhere in the layout via the value of element property <Source>

Do the following:

1. If the layout created in T14 is not open in the Designer: Open 14.vpl.
2. Select any page in <Workarea1>.
3. From the main menu: Select <Edit> / <Find>. The "Find" dialog appears:

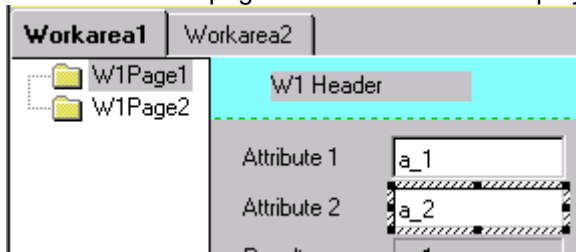


4. For <Name>: Enter <_wh> (name of the <Workarea1> header label).
5. Check <Element name>.
6. Check <Ignore case>.
7. Click <OK>. The workarea header label obtains the focus.



8. Select Workarea2.
9. From the main menu: Select <Edit> / <Find>. The "Find" dialog appears.

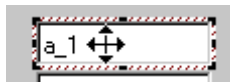
10. For <Name>: Enter <a_2> (value of property <Source> for <Workarea1> page <W1Page1> <Attribute 2> EditField).
11. Check <Attribute name>.
12. Click <OK>. The page with the element is displayed and the element has the focus.



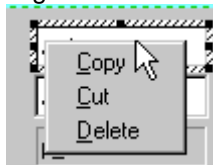
T16. Cut / Copy / Paste

Designer provides standard functions for elements such as copy, cut, paste, and delete. Do the following:

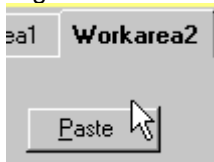
1. If the layout created in T14 is not open in the Designer: Open 14.vpl.
2. In <Workarea1> page <W1Page1>: Select <Attribute 1> EditField.
3. Place the cursor over the element. Note that the cursor becomes a cross:



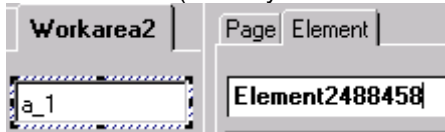
4. Right-click the mouse. A context menu appears:



5. Click <Copy>.
6. Select <Workarea2>.
7. Right-click in the workarea. A context menu appears:



8. Click <Paste> (the only available selection). A copy of the element appears.



Note: The copied element has a new name. An element should be renamed after being copied.

Do not save the layout.

T17. Groups

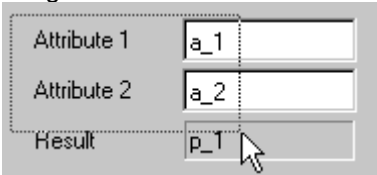
Designer provides extensive functionality for working with groups of elements. Do the following:

1. If the layout created in T14 is not open in the Designer: Open 14.vpl.

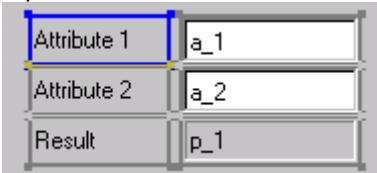
- In <Workarea1> page <W1Page1>: Press and hold the left mouse button to the upper left of the Label <Attribute 1> (not on the label). A small dotted square appears:



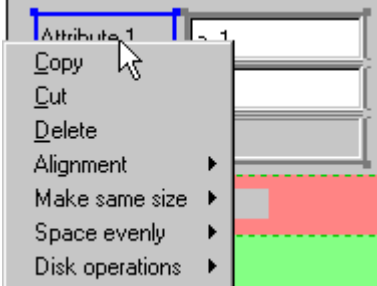
- Drag the mouse until it is over the <p_1> ResultField. Note the dotted square:



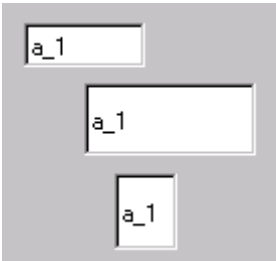
- Release the left mouse button. Note that all elements within or partially within the dotted square are selected. These elements constitute a group:



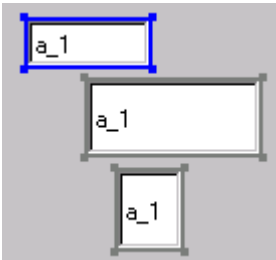
- Right-click over any element in the group. The context menu contains the options for single elements and also the special options for groups:



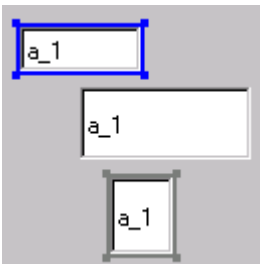
- Create 3 EditElement's in <Workarea2> as shown below:



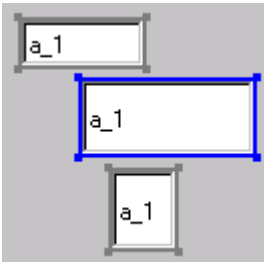
- Select the group. Note that the upper left element has a blue frame, indicating that this element is the selected element within the group.




- Press and hold the Ctrl key.
- Click on the middle EditField. The element is no longer a member of the group:

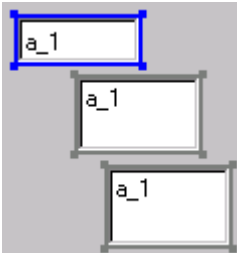



- Click on the middle EditField again (with the Ctrl key still held). The element is now the selected member of the group:

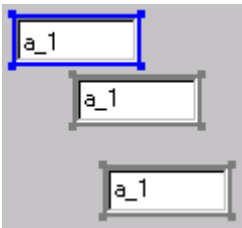



- Reselect the upper EditField in the group.

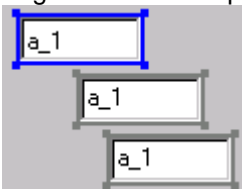
- Click on the <Make all elements the same width> icon (). Note that all of the elements now have the same width as the selected element in the group:




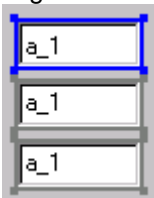
- Click on the <Make all elements the same height> icon (). Note that all of the elements now have the same height as the selected element in the group:




- Click on the <Spread vertical evenly> icon (). Note that the distance between the top edges of the lower pair of elements is the same as that between the upper pair of elements:




- Click on the <Align group to left> icon (). Note that the left edges of all elements are aligned vertically with the left edge of the selected element in the group:



- Select the <Save marked group to disk> icon (). The <Save as> dialog appears.

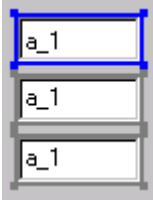
- Save the file as 17.vpg (.vpg file details).

- Delete the group.

- Select the <Read group from disk and copy it to clipboard> icon (). The <Open> dialog appears.

- Open 17.vpg.

- Paste in <Workarea2>.

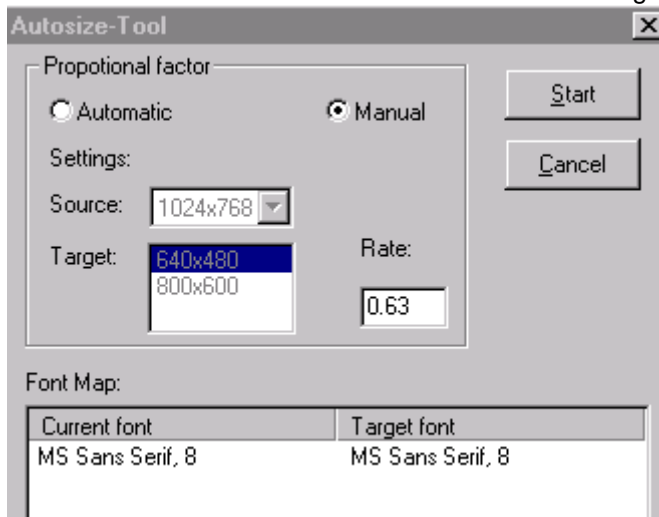


- Delete the group.
Do not save the layout.

T18. Autosize

The entire contents of the layout can be scaled to fit certain screen sizes.
Do the following:

- If the layout created in T14 is not open in the Designer: Open 14.vpl.
- Select <Tools> / <Autosize>. The Autosize-Tool dialog appears.



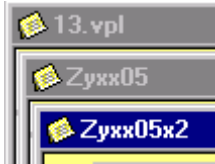
- For proportional factor: Select <Manual>.
- Set <Rate> = 0.5.
- Click <Start>. A new layout is created whose name = (application name) + "x05".



Note that this new layout has been resized with a proportionality factor of 0.5



- Resize the layout with proportional factor 2. A new layout is created whose name = (application name) + "x05x063".



Note that the layout is the original size.

7. Close the 2 new layouts (do not save).
Do not save the original layout.

Multiple layouts/models

It is possible to open a layout from another layout.

A layout can have only a single model. However, that model can include other models whose attributes and properties are available to the the model (and thus the layout).


T19. Multiple layouts

A sublayout can be opened from the same layout using a PushButton element.

The sublayout can define a new "session" or use the same session as the calling layout. If a new session is defined, then values entered for attributes in the sublayout have no effect on the values of the same attributes in the calling layout. Likewise if no new session is defined, then values entered for attributes in the sublayout become the values for the same attributes in the calling layout.

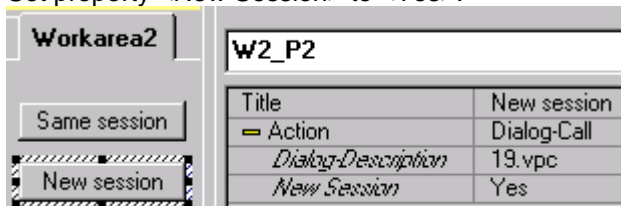
In this example the sublayout will be the same layout as the calling layout (normally the layouts are different).

Do the following:

1. If the layout created in T14 is not open in the Designer: Open 14.vpl.
2. Add a Pushbutton () to <Workarea2>.
3. Set Pushbutton <Name> = <W2_P1>.
4. Set property <Title> = <Same session>.
5. Set property <Action> = <Dialog-Call>.
6. Set property <Dialog-Description> = <19.vpc> (19.vpc will be created when you save this layout). IMPORTANT: Note that the runtime layout (not the design-time) is specified.
7. Set property <New Session> = <No>.

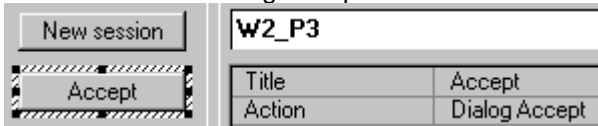


8. Add a Pushbutton to <Workarea2>.
9. Set <Name> = <W2_P2>.
10. Set <Title> = <New session>.
11. Set property <Action> = <Dialog-Call>.
12. Set property <Dialog-Description> = <19.vpc>.
13. Set property <New Session> to <Yes>.

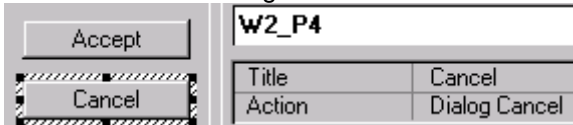


14. Add a Pushbutton to <Workarea2>.
15. Set <Name> = <W2_P3>.
16. Set <Title> = <Accept>.

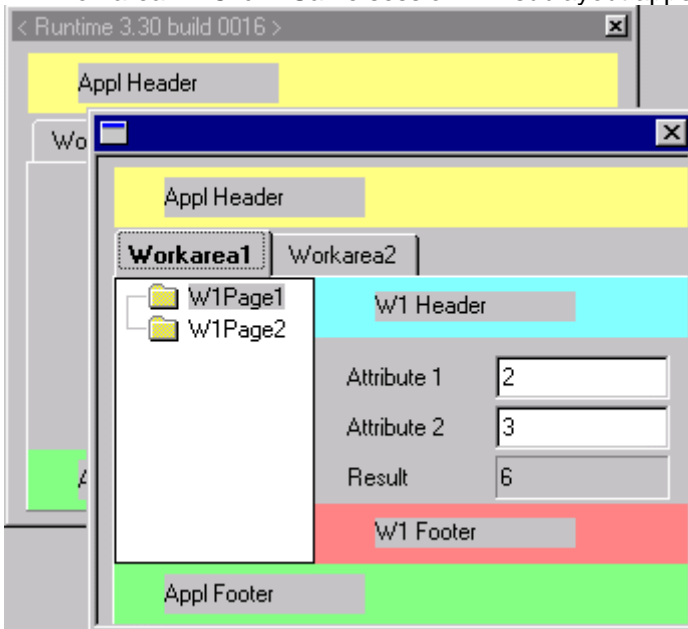
17. Set <Action> = <Dialog Accept>.



18. Add a Pushbutton to <Workarea2>.
19. Set <Name> = <W2_P4>.
20. Set <Title> = <Cancel>.
21. Set <Action> = <Dialog Cancel>.

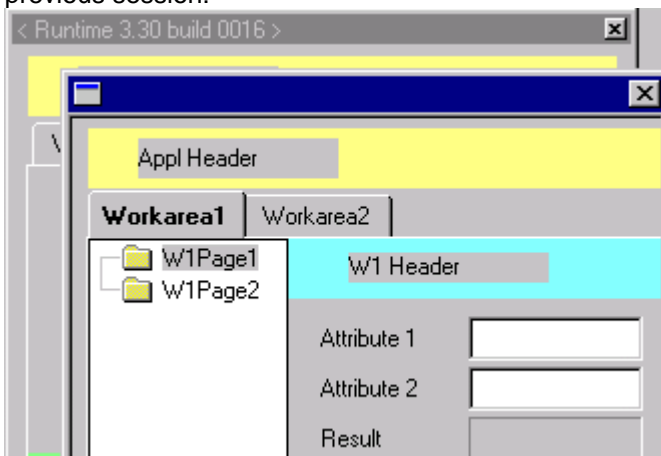


22. Save the layout as 19.vpl.
23. Test the layout.
24. For <Attribute 1>: Enter 2.
25. For <Attribute 2>: Enter 3.
26. In <Workarea2>: Click <Same session>. A sublayout appears:



Note that the values entered in the previous "session" also appear in the new session.

27. For <Attribute 1>: Enter 4.
28. For <Attribute 2>: Enter 5.
29. In <Workarea2>: Select <Accept>. The new session dialog disappears.
In <Workarea1>: Note that <Attribute 1> = 4 and <Attribute 2> = 5.
30. In <Workarea2>: Click <New session>. A sublayout appears without the values entered in the previous session:



31. For <Attribute 1>: Enter 6.
32. For <Attribute 2>: Enter 7.
33. In <Workarea2>: Select <Accept>. The new session dialog disappears.
In <Workarea1>: Note that <Attribute 1> = 4 and <Attribute 2> = 5 (the values have not changed).
34. Test the Accept and Cancel buttons with a Same Session.

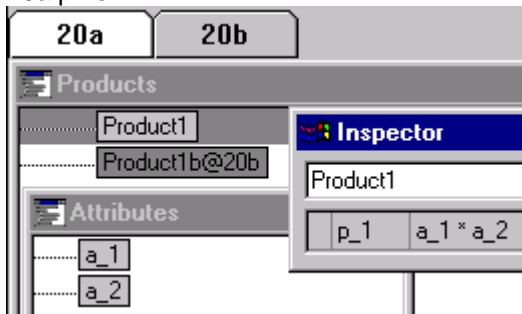
T20. Multiple models

Only a single model can be connected to the layout (specified with property <Product model>). However, this model can contain links to other models, effectively giving the layout access to multiple models.

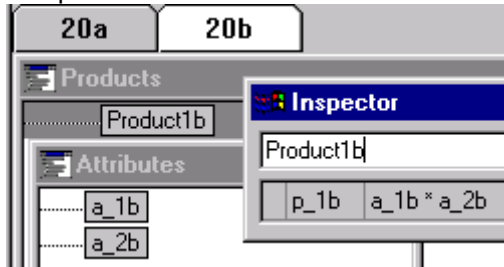
Do the following:

1. If the layout created in T19 is not open in the Designer: Open 19.vpl.
2. Set application property <Product model> = <20a.vpm>. 20a.pms is the same as the previous model, except that it "includes" model 20b.pms.

20a.pms:

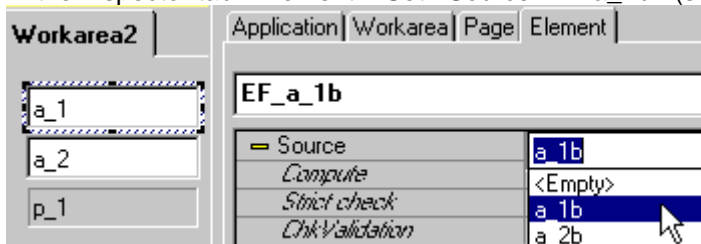


20b.pms:



Note: 20a.pms must have the link ("Product1b@20b") in order to access the properties in 20b.pms. For more information about model inclusion, consult the Workbench documentation.

3. Copy the labels, EditFields and ResultField from <Workarea1> to <Workarea2>.
4. Append the letter "b" to the names and title for each copied element (ie, <EF_a_1> becomes <EF_a_1b>).
5. Open 20a.pms in the Workbench.
6. Select tab <20b> in the Workbench.
7. In Designer: Select <Options> / <Workbench synchronization>.
8. Select the EditField for <a_1b>.
9. In the Inspector tab <Element>: Set <Source> = <a_1b> (select from the drop-down list):



10. For the second EditField: Set <Source> = <a_2b>.

- For the ResultField: Set <Source> = <p_1b> (remember that this value must be typed in, not selected from the drop-down list).

Attribute 1b	a_1b
Attribute 2b	a_2b
Resultb	p_1b

- Save the layout as 20.vpl.
- Test the layout.
- In Workarea2: Enter 3 and 7 for the attributes.
- Click to change the focus. Note that the result was calculated in the included model and displayed in the layout:

Workarea1	Workarea2
Attribute 1b	3
Attribute 2b	7
Resultb	21

External DLL

A layout can utilize functionality provided by an external DLL.

T21. Calling a DLL from the layout

A DLL can be called from the layout.

Do the following:

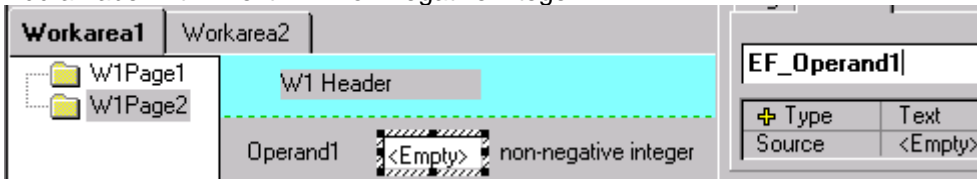
- If the layout created in T20 is not open in the Designer: Open 20.vpl.
- Open the file 21.ini. 21.ini contains the following code:


```

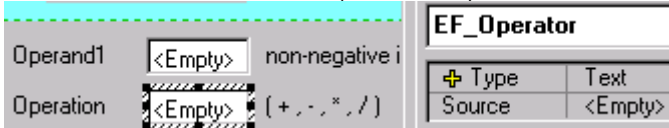
;Configuration file for DLL-call
;=====
[common]
; 32-Bit DLL path (relative to layout) / filename
DLL=21.dll
Method=Rechnen
; 16-Bit DLL path (relative to layout) / filename
DLL16=21.dll
Method16=Rechnen
;=====
[input]
; (DLL-call input)=(layout input element name)
0=$hwnd
1=EF_Operand1
2=EF_Operand2
3=EF_Operator
;=====
[output]
; (DLL-call result)=(layout display element name)
0=EF_Result
;=====

```
- Save 21.ini.
- In layout <Workarea1> page <W1Page2>: Add a label with <Text> = <Operand1>.
- Add a text EditField with <Name> = <EF_Operand1>.

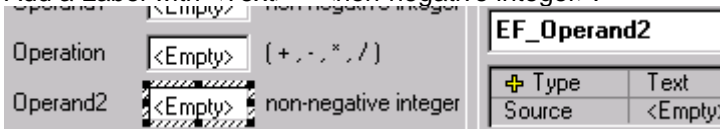
6. Add a Label with <Text> = <non-negative integer>.



7. Add a Label with <Text> = <Operator>.
8. Add a text EditField with <Name> = <EF_Operator>.
9. Add a Label with <Text> = <(+ , - , * , /)>.



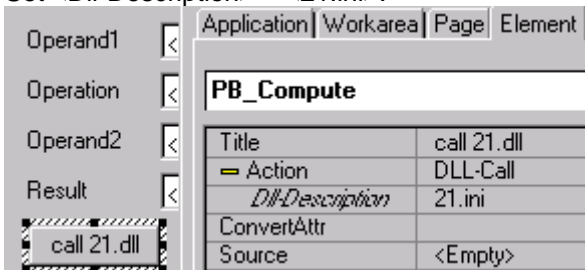
10. Add a Label with <Text> = <Operand>.
11. Add a text EditField with <Name> = <EF_Operand2>.
12. Add a Label with <Text> = <non-negative integer>.



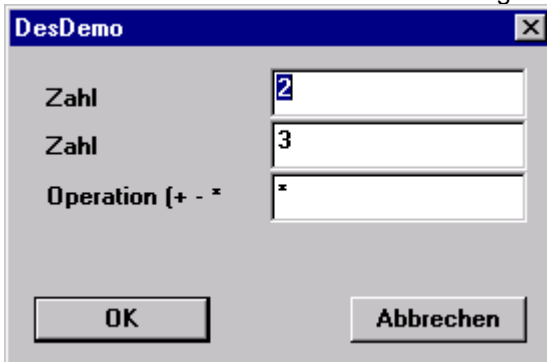
13. Add a Label with <Text> = <Result>.
14. Add a text EditField with <Name> = <EF_Result>.



15. Add a Pushbutton with <Name> = <PB_Compute>.
16. Set <Title> = <call 21.dll>.
17. Set <Action> = <DLL-call>.
18. Set <Dll-Description> = <21.ini>.



19. Save the layout as 21.vpl.
20. Test the layout.
21. For Operand1: Enter 2.
22. For Operator: Enter *.
23. For Operand2: Enter 3.
24. Click Pushbutton "call 21.dll". The following dialog appears:



25. Click OK. The result is displayed:

Operand1	2
Operation	*
Operand2	3
Result	6

Documenting a layout

Layouts can be complicated, and therefore its important to be able to somehow document the layout. Designer supports the following documentation functions:

- Comments can be included in the layout for each component.
- Reports about the content can be generated.
- Screenshots can be generated for each layout page.

T22. Commenting layout components

Comments can be included in the layout for any layout components.

Do the following:

1. If the layout created in T21 is not open in the Designer: Open 21.vpl.
2. Set application property <Version info> = <Yes>. Note that the subproperty <Version> appears:

Application	Workarea	Page
Zyx		
Version info	Yes	
Version		

3. Click in to the right of <Version>. A button appears.

Version info	Yes	
Version		
Title		

4. Click on the button. The <Version information> dialog appears.
5. Enter commentary as shown:

Version information			
Title	Application title	Version	v 1.0
Description			
Description of application.			
Created by	Creator	Created date	Date created
Last update by	Last updater	Last updated date	Date of last update
Last updated description			
Description of last update			

- Click <OK>. Note the text from <Version> in the Inspector:

Version info	Yes
Version	v 1.0

- Optional: Comment any other components (workareas, pages, elements).
- Save the layout as 22.vpl.

T23. Generating layout report

2 report formats are available:

- Table
- Text

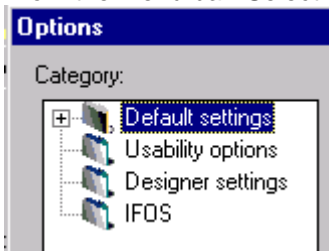
The content of the report can also be selected.

More detailed info about reports.

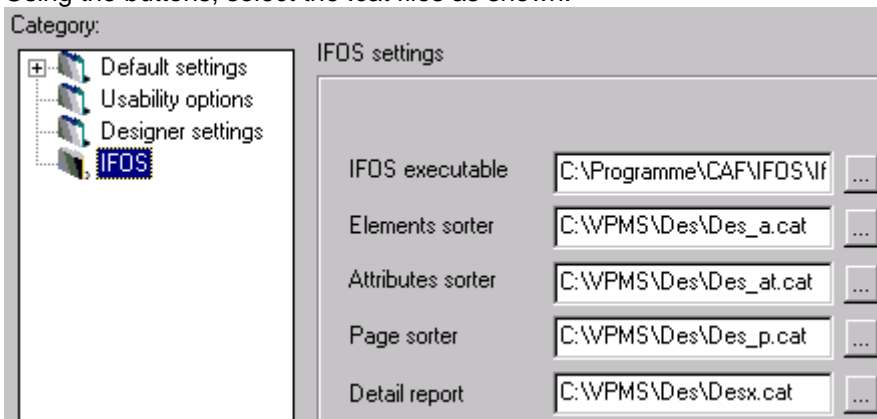
In this chapter you will create report in table and text format.

Do the following:

- If the layout created in T22 is not open in the Designer: Open 22.vpl.
- From the menu bar: Select <Options> / <Options...>. The <Options> dialog appears.



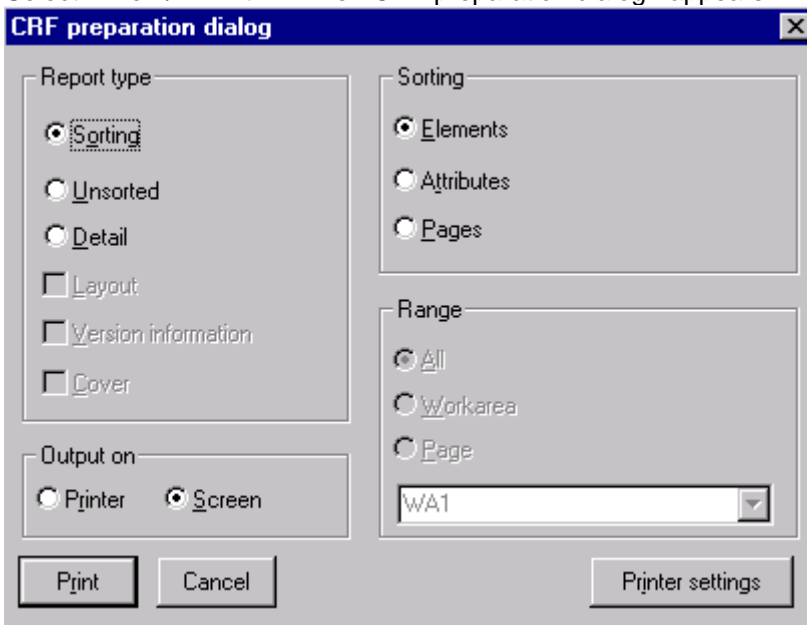
- Select category <IFOS>.
- Using the buttons, select the .cat files as shown:



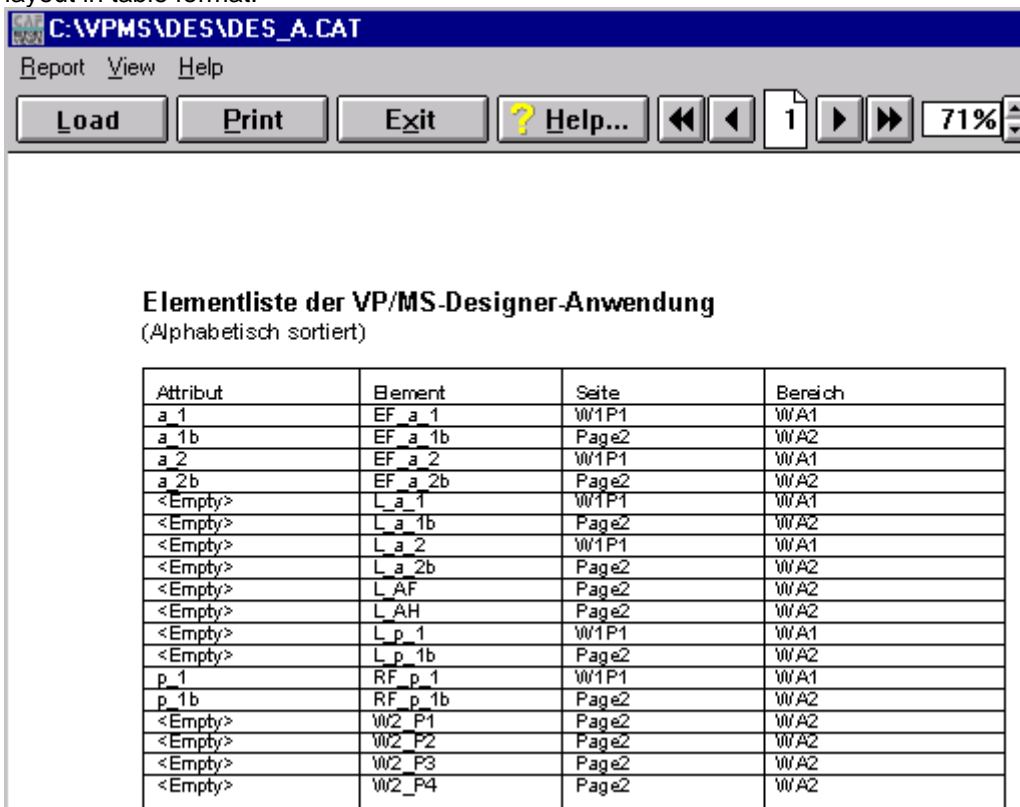
Note: The location of the .cat files may be different on your computer. The full directory path and filename is required.

- Click <OK>.

6. Select <File> / <Print...>. The <CRF preparation dialog> appears:



7. Select <Report type>, <Sorting>, and <Output> on as shown in the dialog above.
8. Click <Print>. The IFOS ReportViewer appears with a report describing the contents of the layout in table format:



9. Close the ReportViewer.
10. In Designer: Select <File> / <Print...>.
11. In the <CRF preparation dialog>: Select <Report type> = <Detail>, check <Version information>, select <Range> = <All>, <Output on> = <Screen>.
12. Click <Print>. The IFOS ReportViewer appears with a report (approximately 12 pages) describing the contents of the layout in text format. The first page should be similar to the

following (note the version information (comments)):

VP/MS-Designer-Anwendung C:\VPMS\

Anwendungseigenschaften:

Titel:	
Breite:	294
Höhe:	233
Format:	Individual
Produktmodell:	15a.vpm
Stil:	Notebook
TAB-Font:	
Berechnung:	Automatic
Startattribut:	No
Startwert:	
Standardschrift:	MS Sans Serif, 8
Farbe aktives Element:	(255,255,255)
Externe DB	No

Versionsinformationen der Anwendung:

Titel:	Application title
Versionsnummer:	v 1.0
Erstellt von:	Creator
Erstellt am:	Date created
Beschreibung:	Description of application.
Letzte Änderung von:	Last updatator
Letzte Änderung am:	Date of last update
Beschreibung letzte Änderung:	Description of last update

13. Close the ReportViewer.
14. Save the layout as 23.vpl.

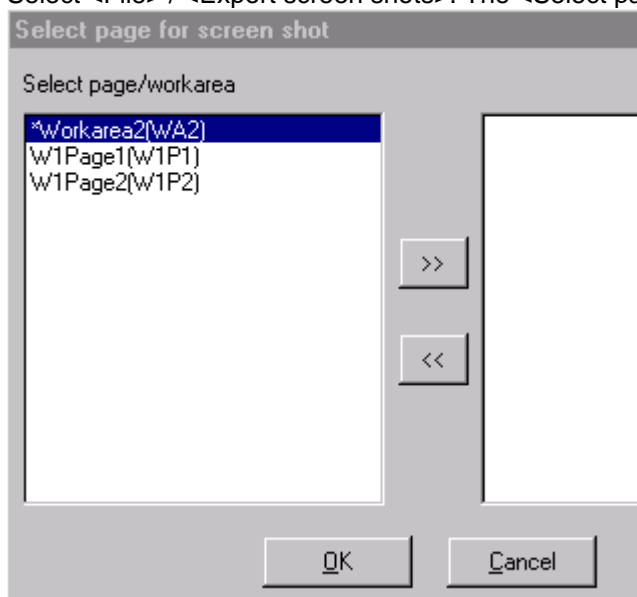
T24. Creating layout page screenshot

Screenshots (.bmp files) can be created for any of the layout pages. A page screenshot includes the entire layout as it would appear with that page selected.

The .bmp files are normally saved to the same directory as the .cat files.

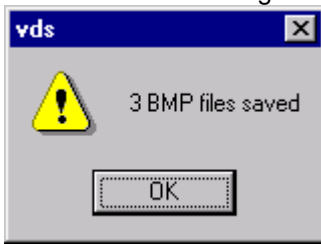
Do the following:

1. If the layout created in T23 is not open in the Designer: Open 23.vpl.
2. Select <File> / <Export screen shots>. The <Select page for screen shot> dialog appears:



All of the layout pages are listed in the left window. The currently visible page is marked with an asterisk (*).

3. Select (move to the right window) all pages using the ">>" key.
4. Click <OK>. A message about the created screenshots appears:



The bmp files (scr_W1P1.bmp, scr_W1P2.bmp, and scr_WA2().bmp) were saved in the directory where the .cat files are located.

5. Do not save the layout.

T25. Compare layouts

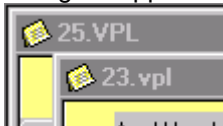
A report can be generated that describes the differences between 2 layouts.

Do the following:

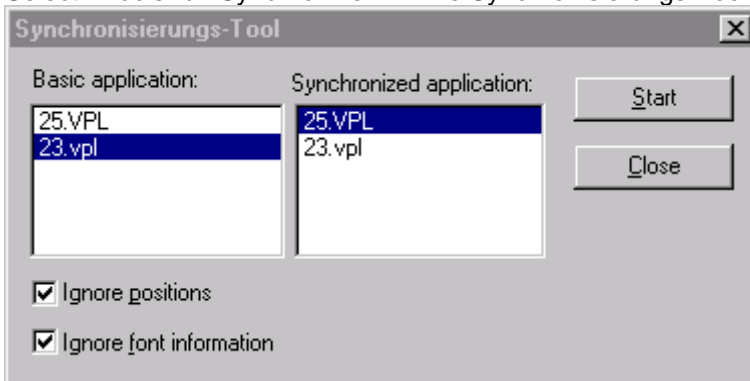
1. If the layout created in T23 is not open in the Designer: Open 23.vpl.
2. In <Workarea1> page <W1Page2>: Add a Label with <Name> = <L_new> and <Title> = <Labelnew>.



3. Save the layout as 25.vpl.
4. Select <File> / <Open> and open 23.vpl (there are now 2 Designer layouts open in a single Designer application).

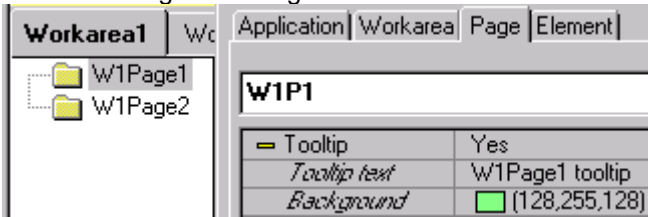


5. Select <Tools> / <Synchronize...>. The Synchronisierungs-Tool appears:

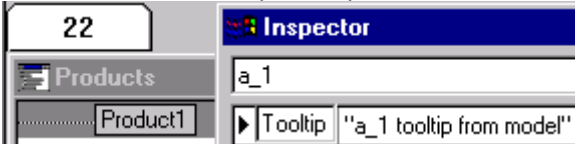


6. Select 23.vpl as the Basic application and 25.vpl as the synchronized application.

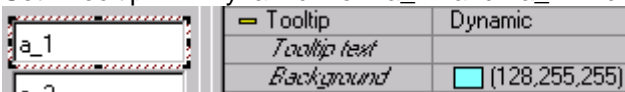
8. Set <Tooltip> = <Yes>. The tooltip subproperties appear.
9. Set <Tooltip text> = <W1Page1 tooltip>.
10. Select <Background> = green.



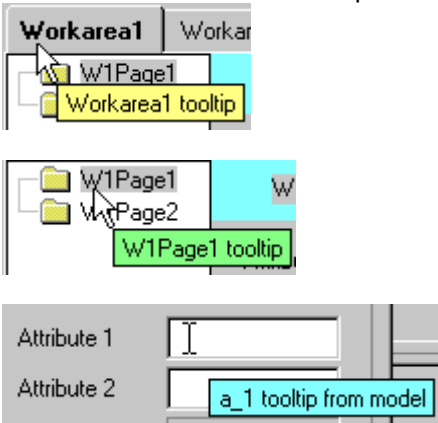
11. Set the model as 26.vpm. 26.pms contains the tooltip for <a_1> and <a_2>.



12. Set <Tooltip> = <Dynamic> for <a_1> and <a_2> EditField's.



13. Save the layout as 26.vpl.
14. Test the layout.
15. Place the cursor over the components to display the tooltips:



Datatypes

This part introduces the data types supported by Designer.

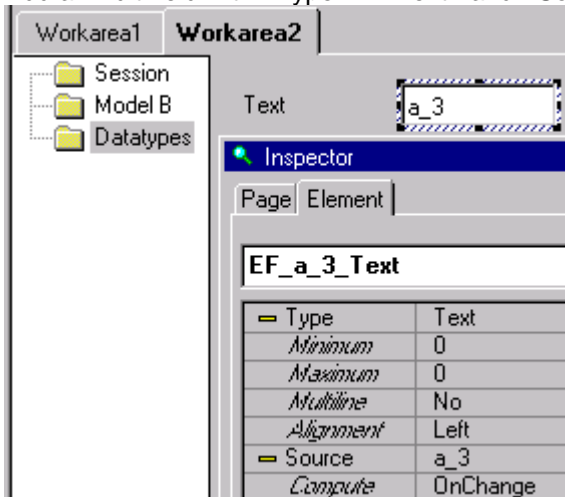
T27. Datatype: Single-line text

Single-line text is the most common data type. You have already used single-line text EditField's in the previous examples for entering <a_1> and <a_2> values.

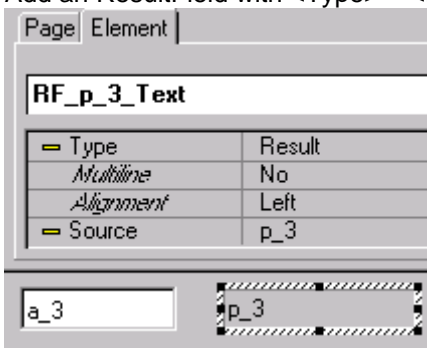
Do the following:

1. If the layout created in T26 is not open in the Designer: Open 26.vpl.
2. Change the Product model to 27.vpm.
3. In <Workarea2>: Add page <Datatypes>.
4. Add a Label with <Title> = <Text>.

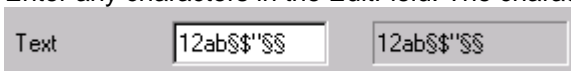
5. Add an EditField with <Type> = <Text> and <Source> = <a_3>.



6. Add an ResultField with <Type> = <Result> and <Source> = <p_3>.



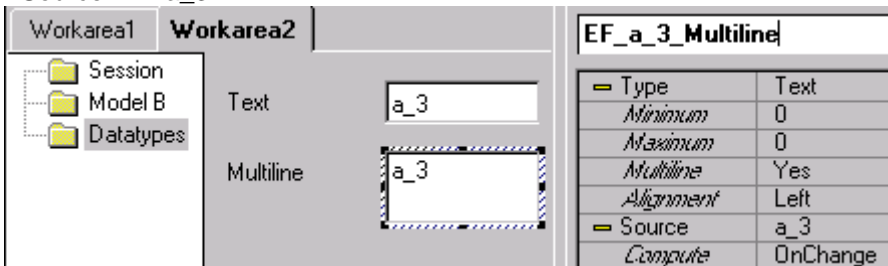
7. Save the layout as 27.vpl.
8. Test the layout.
9. Enter any characters in the EditField. The characters are displayed in the ResultField.



T28. Datatype: Multiline text

An EditField can be used to enter (or display) multiline text.
Do the following:

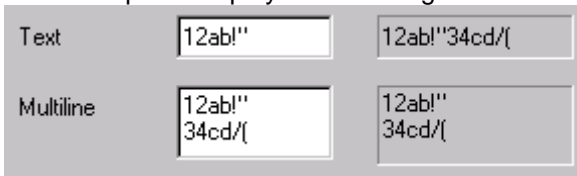
1. If the layout created in T27 is not open in the Designer: Open 27.vpl.
2. Add a Label with <Title> = <Multiline>.
3. Add an EditField with <Type> = <Text>, <Multiline> = <Yes>, <Compute> = <OnChange> and <Source> = <a_3>.



4. Add an ResultField with <Type> = <Result>, <Multiline> = <Yes> and <Source> = <p_3>.



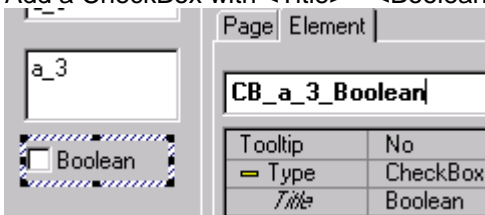
5. Save the layout as 28.vpl.
6. Test the layout.
7. Enter multiline data in the multiline EditField (press the Return key to start a new line). Note how the input is displayed in the single-line EditField and in the ResultField's.



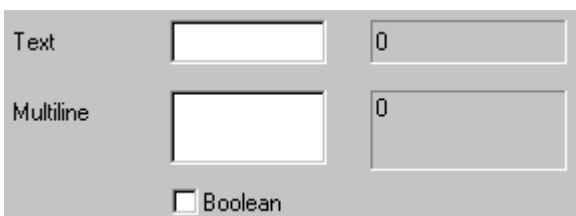
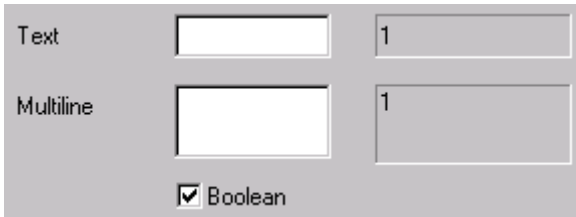
T29. Datatype: Boolean

A Checkbox can be used to enter a boolean value.
Do the following:

1. If the layout created in T28 is not open in the Designer: Open 28.vpl.
2. Add a CheckBox with <Title> = <Boolean> and <Source> = <a_3>.



3. Save the layout as 29.vpl.
4. Test the layout.
5. Check and uncheck the checkbox. Note that the boolean value appears in the ResultFields.



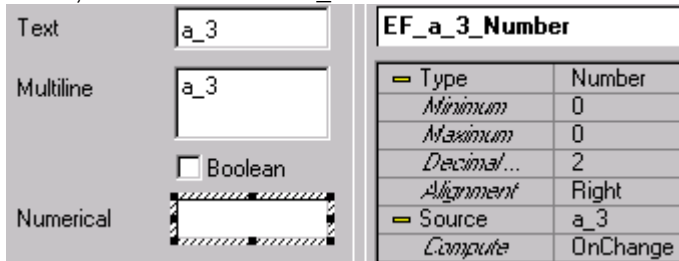
T30. Datatype: Number

An EditField can be used to enter (or display) numerical values with a fixed number of digits after the decimal point.

Do the following:

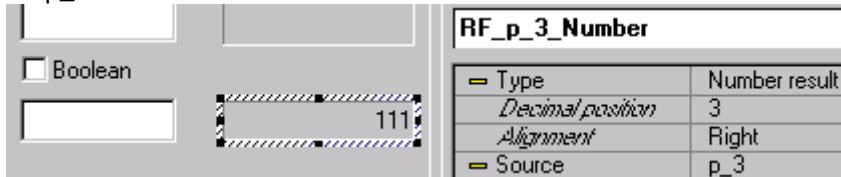
1. If the layout created in T29 is not open in the Designer: Open 29.vpl.

2. Add a Label with <Title> = <Number>.
3. Add an EditField with <Type> = <Number>, <Compute> = <OnChange>, <Decimal position> = <2>, and <Source> = <a_3>.



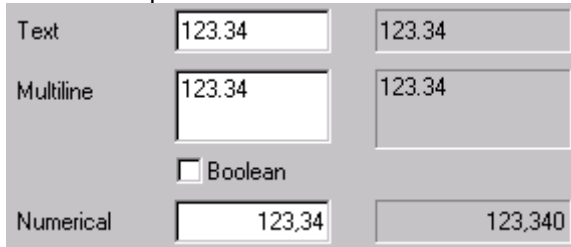
EF_a_3_Number	
Type	Number
Minimum	0
Maximum	0
Decimal...	2
Alignment	Right
Source	a_3
Compute	OnChange

4. Add an ResultField with <Type> = <Number result>, <Decimal position> = <3> and <Source> = <p_3>.



RF_p_3_Number	
Type	Number result
Decimal position	3
Alignment	Right
Source	p_3

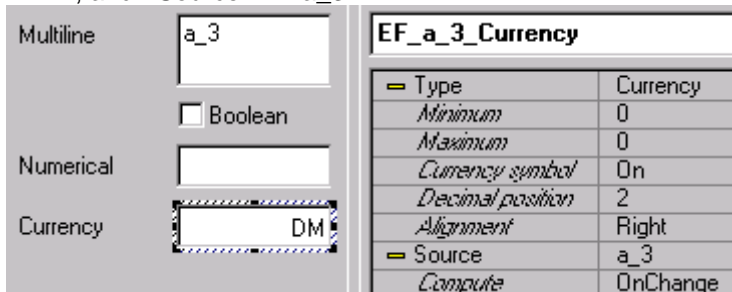
5. Save the layout as 30.vpl.
6. Test the layout.
7. Enter in the NumberField 123,3456. Note that the input is limited to 2 decimal positions after the decimal point. Note also the values shown in the other elements.



T31. Datatype: Currency

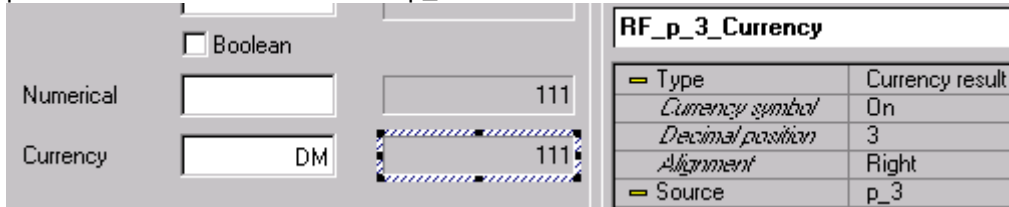
An EditField can be used to enter (or display) currency value with/without a currency unit. Do the following:

1. If the layout created in T30 is not open in the Designer: Open 30.vpl.
2. Add a Label with <Title> = <Currency>.
3. Add an EditField with <Type> = <Currency>, <Compute> = <OnChange>, <Decimal position> = <2>, and <Source> = <a_3>.



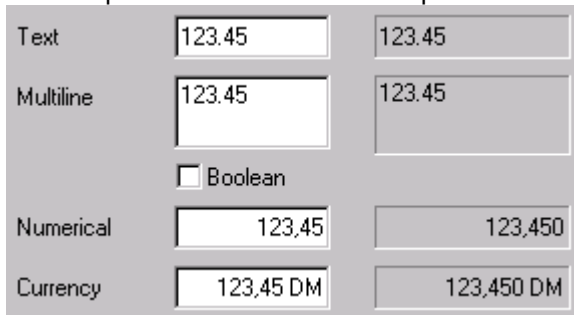
EF_a_3_Currency	
Type	Currency
Minimum	0
Maximum	0
Currency symbol	On
Decimal position	2
Alignment	Right
Source	a_3
Compute	OnChange

4. Add an ResultField with <Type> = <Currency result>, <Currency symbol> = <On>, <Decimal position> = <3> and <Source> = <p_3>.



RF_p_3_Currency	
Type	Currency result
Currency symbol	On
Decimal position	3
Alignment	Right
Source	p_3

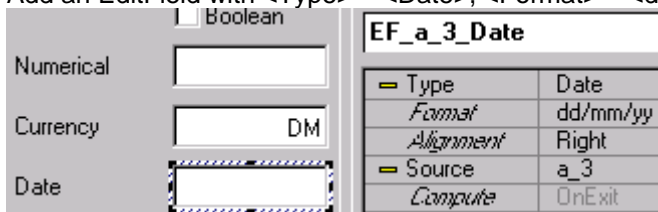
5. Save the layout as 31.vpl.
6. Test the layout.
7. Enter in the CurrencyField 123,3456.
8. Click to change the focus and display the currency symbol. Note that the input is limited to 2 decimal positions after the decimal point. Note also the values shown in the other elements.



T32. Datatype: Date

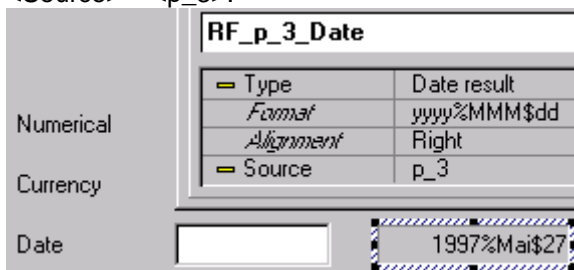
A DateField can be used to enter (or display) Date values of various formats. Do the following:

1. If the layout created in T31 is not open in the Designer: Open 31.vpl.
2. Add a Label with <Title> = <Date>.
3. Add an EditField with <Type> = <Date>, <Format> = <dd/mm/yy> and <Source> = <a_3>.



EF_a_3_Date	
Type	Date
Format	dd/mm/yy
Alignment	Right
Source	a_3
Compute	OnExit

4. Add an ResultField with <Type> = <Date result>, <Format> = <yyyy%MMM\$dd> and <Source> = <p_3>.



RF_p_3_Date	
Type	Date result
Format	yyyy%MMM\$dd
Alignment	Right
Source	p_3

5. Save the layout as 32.vpl.
6. Test the layout.
7. Enter in the DateField 010299 (Feb 01, 99).

- Click to change the focus. Note the values shown in the other elements.

Text	01.02.2099	01.02.2099
Multiline	01.02.2099	01.02.2099
	<input type="checkbox"/> Boolean	
Numerical	1,02	1,02.
Currency	1,02 DM	1,02. DM
Date	01/02/99	2099%Feb\$01

T33. Datatype: Mask

A mask can be used to specify a special format for valid data.

Do the following:

- If the layout created in T32 is not open in the Designer: Open 32.vpl.
- Add a Label with <Title> = <Mask>.
- Add an EditField with <Type> = <Mask>, <Mask> = <ULLL##> and <Source> = <a_3>.

Date		Type	Mask
Mask	a_3	Mask	ULLL##
		Alignment	Right
		Source	a_3

- Add an ResultField with <Type> = <Mask result>, <Mask> = <UUUU##> and <Source> = <p_3>.

		Type	Mask result
Currency		Mask	UUUU##
		Alignment	Right
Date		Source	p_3
Mask	a_3		p_3

- Save the layout as 33.vpl.
- Test the layout.
- Enter in the MaskField abcde12345.
- Click to change the focus. Note the values shown in the other elements.

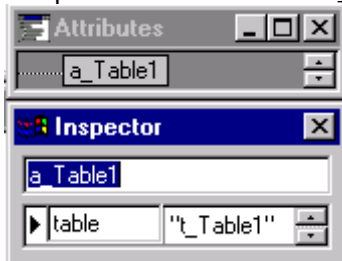
Text	Abcd12	Abcd12
Multiline	Abcd12	Abcd12
	<input type="checkbox"/> Boolean	
Numerical	Abc.d12,00	Abc.d12,000
Currency	Abc.d12,00	Abc.d12,000 DM
Date	__/__/__	___%__\$
Mask	Abcd12	ABCD12

T34. Datatypes: Arrays: Tables

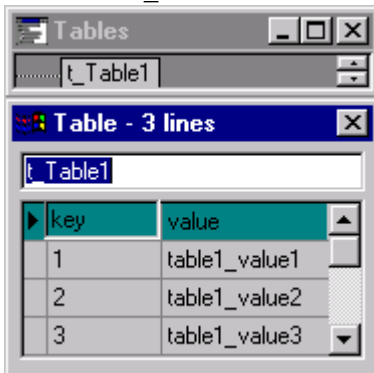
This section demonstrates how a model table can be displayed in the layout and how values from the table can be selected.

Do the following:

1. If the layout created in T33 is not open in the Designer: Open 33.vpl.
2. Change the Product model to 34.vpm.
34.vpm contains attribute <a_Table1> with property <table> = <"t_Table1">:

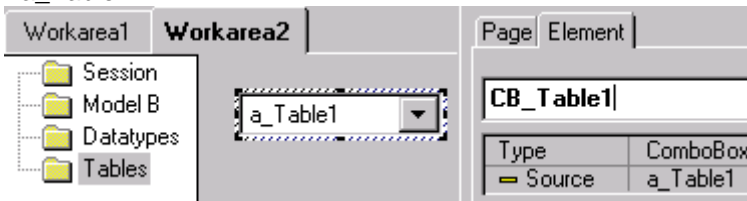


and table <t_Table1> with columns <key> and <value> with 3 rows of data:

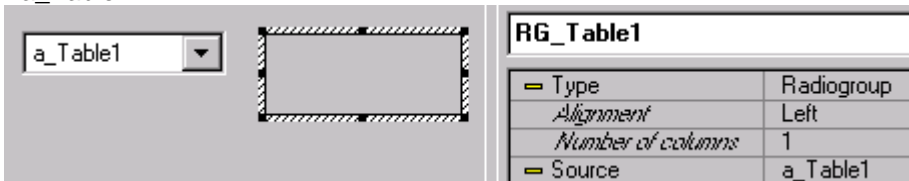


Details about this model.

3. In <Workarea2>: Add a page with <Name> = <PTable> and <Title> = <Tables>.
4. In Page <PTable>: Add a Combobox with <Name> = <CB_Table1> and <Source> = <a_Table1>:



5. In Page <PTable>: Add a Combobox with <Name> = <CB_Table1> and <Source> = <a_Table1>:



6. Save the layout as 34.vpl.
7. Test the layout.
8. Select a value from the ComboBox or RadioGroup. Note that the value is displayed in both elements.



T35. Datatypes: Arrays: Iteration

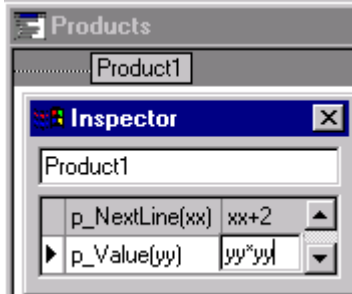
This section demonstrates how an array of data can be generated using iteration. The array is generated with the help of and display by a Grid.

Do the following:

1. If the layout created in T34 is not open in the Designer: Open 34.vpl.
2. Change the Product model to 35.vpm.
35.vpm contains attribute <a_Iteration> and <a_Start>:

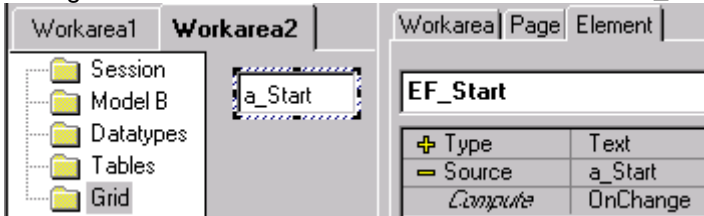


and a next-line property and a value for each iteration:

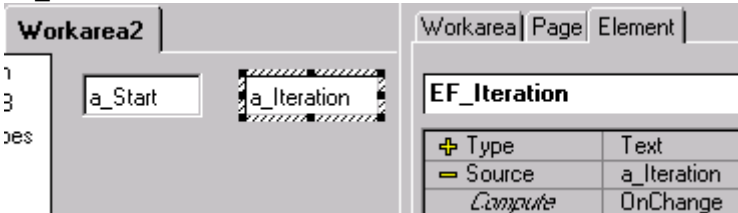


Note: <xx> and <yy> used above could be changed to any string with no effect.
Details about this model.

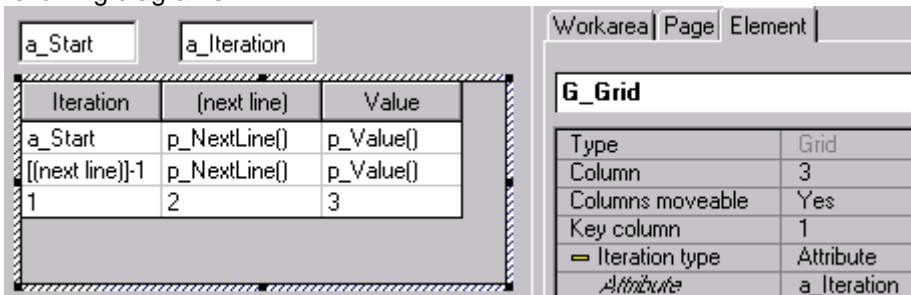
3. In <Workarea2>: Add a page with <Name> = <PGrid> and <Title> = <Grid>.
4. In Page <PGrid>: Add an EditField with <Name> = <EF_Start> and <Source> = <a_Start>:



5. In Page <PGrid>: Add an EditField with <Name> = <EF_Iteration> and <Source> = <a_Iteration>:



6. In Page <PGrid>: Add a <Grid> with name = <G_Grid> and with properties as shown in the following diagrams:



Property for	Column 1
Title	Iteration
Width	70
Usage	Input
Attribute	a_Start
Next line	2
Type	Text
Alignment	Left

▼ Property for	Column 2
Title	(next line)
Width	80
Usage	Output
Property	p_NextLine()
Type	Text
Alignment	Left

▼ Property for	Column 3
Title	Value
Width	70
Usage	Output
Property	p_Value()
Type	Number
Decimalposition	0
Alignment	Left

7. Save the layout as 35.vpl.
8. Test the layout.
9. Set a_Start = 3.
10. Set a_Iteration = 4. Note the resultant Grid.

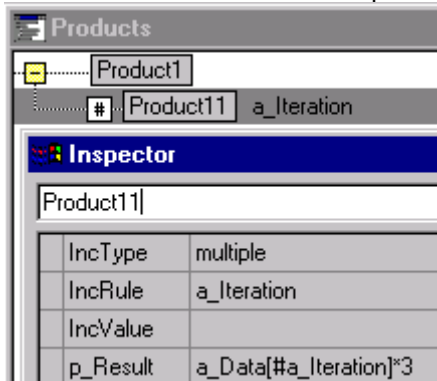
<input type="text" value="3"/>	<input type="text" value="4"/>	
Iteration	(next line)	Value
3	4	9
4	5	16
5	6	25
6	7	36

T36. Datatypes: Arrays: Multiple pages

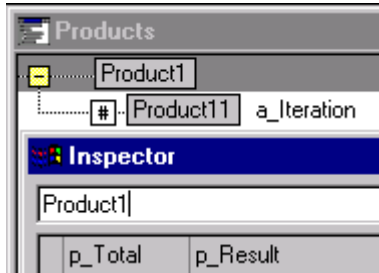
An array of data can be generated using multiple inclusion in the model and multiple subpages in the layout.

Do the following:

1. If the layout created in T35 is not open in the Designer: Open 35.vpl.
2. Change the Product model to 36.vpm.
36.vpm adds the attribute <a_Data>.
and a model subnode with mutple inclusion and property <p_Result>:

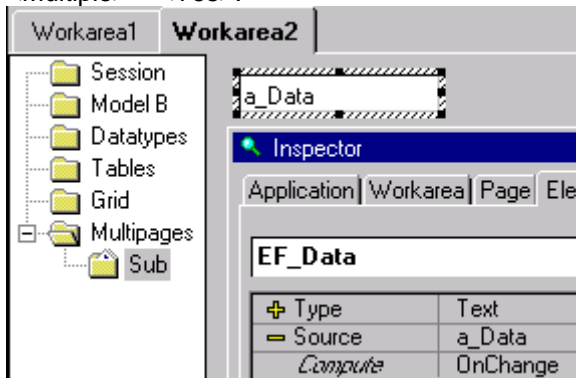


The upper branch also adds property <p_Total>:

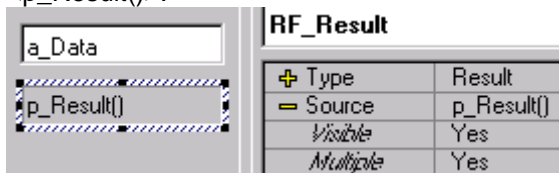


Details about this model.

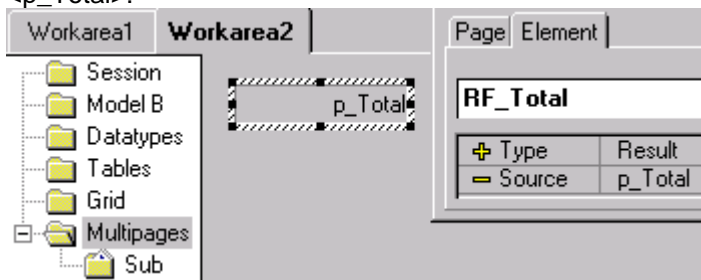
3. In Workarea2: Add a page with <Name> = <PMultipages> and <Title> = <Multipages>.
4. In Workarea2: Add a subpage for page <Multipages> with <Name> = <PSub> and <Title> = <Sub>.
5. In Page <PSub>: Add an EditField with <Name> = <EF_Data> and <Source> = <a_Data> and <Multiple> = <Yes>:



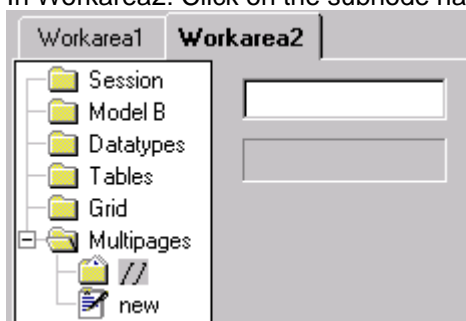
6. In Page <PSub>: Add a ResultField with <Name> = <RF_Result> and <Source> = <p_Result()>:



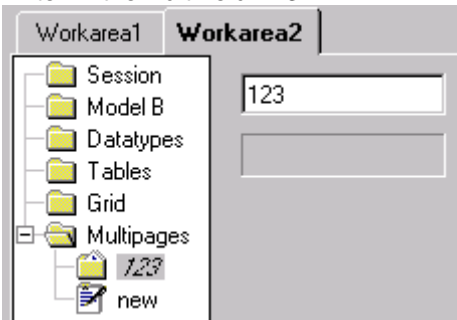
7. In Page <Multipages>: Add a <ResultField> with <Name> = <RF_Total> and <Source> = <p_Total>:



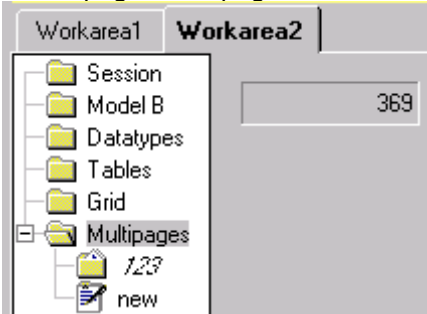
8. Save the layout as 36.vpl.
9. Test the layout.
10. In Workarea2: Click on the subnode named <new>. A new page appears:



11. Enter in the EditField 123:



12. Select page <Multipages>. Note that the result is displayed in the page:



Entering/selecting data

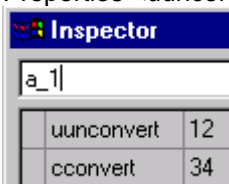
In the previous pages you used layout elements to enter (EditField) and select (ComboBox) data. In this part you will learn more advanced aspects of entering/selecting data.

T37. Conversion list

The values for multiple attributes in a layout can be set (converted) by simply clicking on a PushButton.

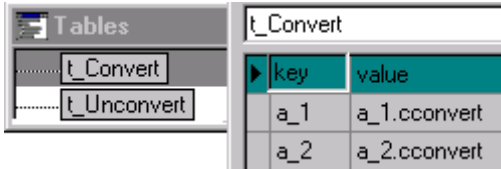
Do the following:

1. If the layout created in T36 is not open in the Designer: Open 36.vpl.
2. Set the model to 37.vpm. 37.pms specifies the following:
Properties <uunconvert> and <cconvert> for attributes <a_1> and <a_2>.

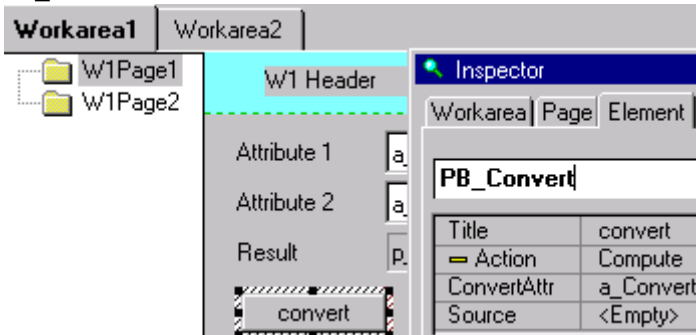


Note: The attribute property names <uunconvert> and <cconvert> could be any non-reserved word.

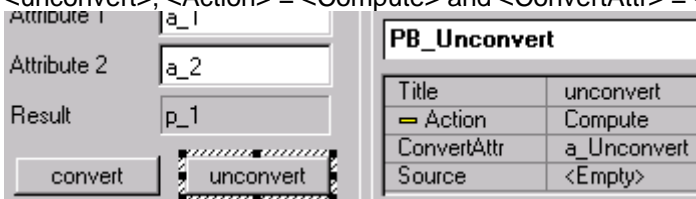




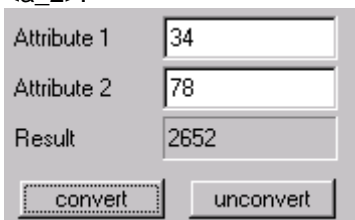
- In the layout: In <Workarea1> <W1Page1>: Add a PushButton with <Name> = <PB_Convert>, <Title> = <convert>, <Action> = <Compute> and <ConvertAttr> = <a_Convert>:



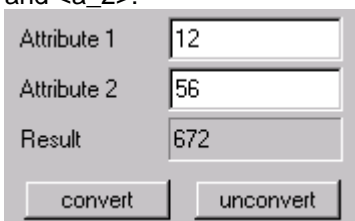
- In <Workarea1> <W1Page1>: Add a PushButton with <Name> = <PB_Unconvert>, <Title> = <unconvert>, <Action> = <Compute> and <ConvertAttr> = <a_Unconvert>:



- Save the layout as 37.vpl.
- Test the layout.
- Click on the <convert> pushbutton. Note that the convert values are entered for <a_1> and <a_2>:



- Click on the <unconvert> pushbutton. Note that the unconvert values are entered for <a_1> and <a_2>:



T38. Dynamic Combobox

The available selections in a ComboBox can be dynamic, which means that the available selections change in the runtime layout depending on some other layout attribute.

Do the following:

1. If the layout created in T37 is not open in the Designer: Open 37.vpl.
2. Change the product model to 38.vpm. 38.pms contains the following:
Attribute <a_Table1> has property <dynamic> = <"a_Table2">. This means that when a change is made (in the layout) to <a_Table1> using a layout component with property <ChkDynamic> = <Yes>: The value for <a_Table2> will be recalculated.

a_Table1	
table	"t_Table1"
dynamic	"a_Table2"

<t_Table1> as in previous layouts:

t_Table1	
key	value
1	table1_value1
2	table1_value2
3	table1_value3

<a_Table2> possible values are defined in <t_Table2>. The property <display> specifies which column in <t_Table2> should be displayed (<chooseable>).

a_Table2	
table	"t_Table2"
display	"chooseable"

<t_Table2> has several values in column <chooseable> for each possible value in column <value>.

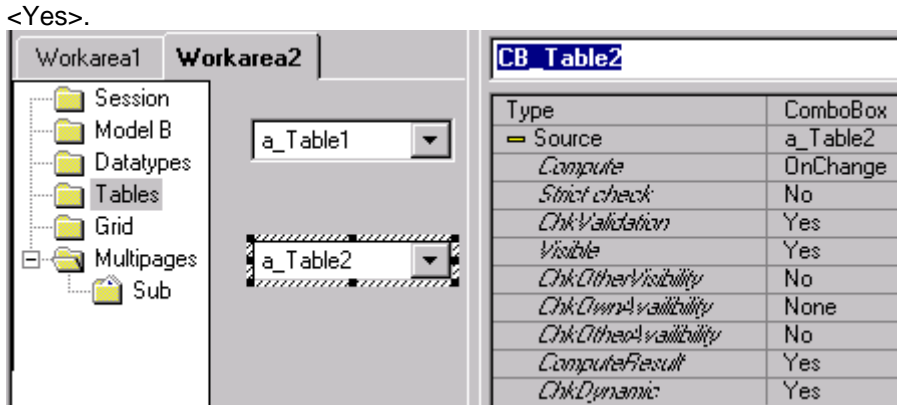
t_Table2		
key	value	chooseable
1	table1_value1	t1v1_a
2	table1_value1	t1v1_b
3	table1_value2	t1v2_a
4	table1_value2	t1v2_b
5	table1_value3	t1v3_a
6	table1_value3	t1v3_b

<a_Table2> property <filter> defines that only those rows in <t_Table2> whose value in column <value> = value in column <value> in table <t_Table1> are to be available (or "chooseable").

a_Table2	
filter	if(strcmp(t_Table2[key];t_Table1[a_Table1])=0;1;0)

For more details, consult the Workbench User's Guide.

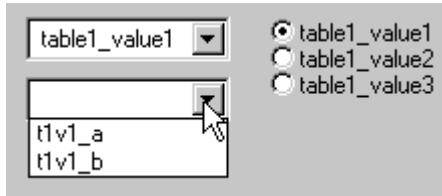
3. In the layout: In <Workarea2> page <Tables>: Add a ComboBox with <Name> = <CB_Table2>, <Source> = <a_Table2>, <Compute> = <OnChange> and <ChkDynamic> =



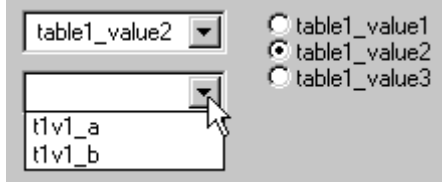
- Set <a_Table1> property <ChkDynamic> = <Yes>.



- Save the layout as 38.vpl.
- Test the layout.
- In <Workarea2> page <Tables>: Select <table1_value1>.
- Click on the lower CheckBox. Note that only the values in <a_Table2> column <chooseable> whose value in column <value> matches <table1_value1> can be selected:



- Select the RadioButton <table1_value2>.
- Click on the lower CheckBox. Note that the available values have not changed:



The value did not change because the RadioGroup attribute property <ChkDynamic> = <No>.

T39. Element order

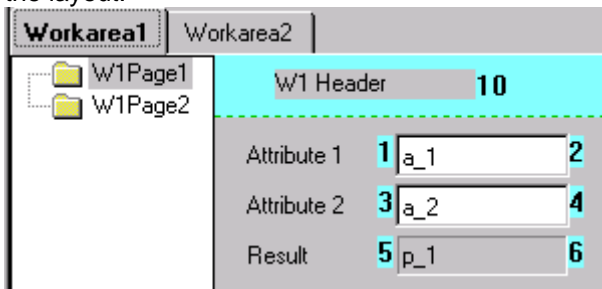
An important aspect in entering data in a layout is the order in which the elements gain the focus when the tab key is pressed. This order can be specified in the Designer.

Do the following:

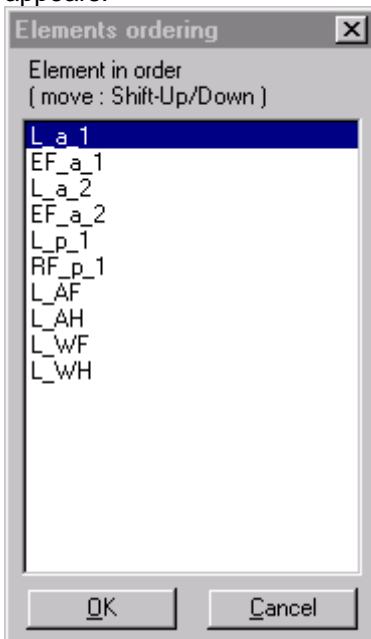
- If the layout created in T38 is not open in the Designer: Open 38.vpl.
- Select <Workarea1> page <W1Page1>.
- In Inspector tab <Page>: Set <Order> = <Individual>.



- From the menu: Select <Options> / <Display order>. The order of the elements is displayed in the layout:

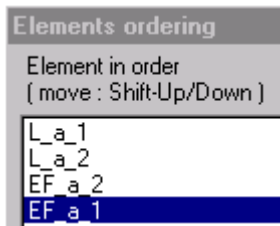


- From the menu: Select <Edit> / <Elements reordering>. The <Elements reordering> dialog appears.

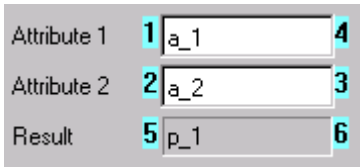


Note that the elements are listed in the order as shown by the numbers in the layout.

- Select <EF_a_1>.
- Press and hold the <Shift> key.
- Click the down arrow key 2 times (until <EF_a_1> is below <EF_a_2> in the dialog).



- Click <OK>. Note the new order of the elements:



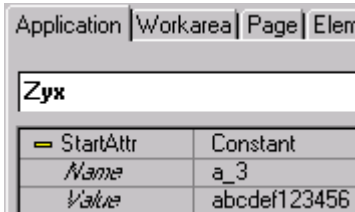
- Save the layout as 39.vpl.
- Test the layout.
- Click on the <Workarea1> tab.
- Click the <Tab> key. <W1Page1> obtains the focus.
- Click the <Tab> key. The EditField for <Attribute 2> obtains the focus.
- Click the <Tab> key. The EditField for <Attribute 1> obtains the focus.

T40. Start attribute

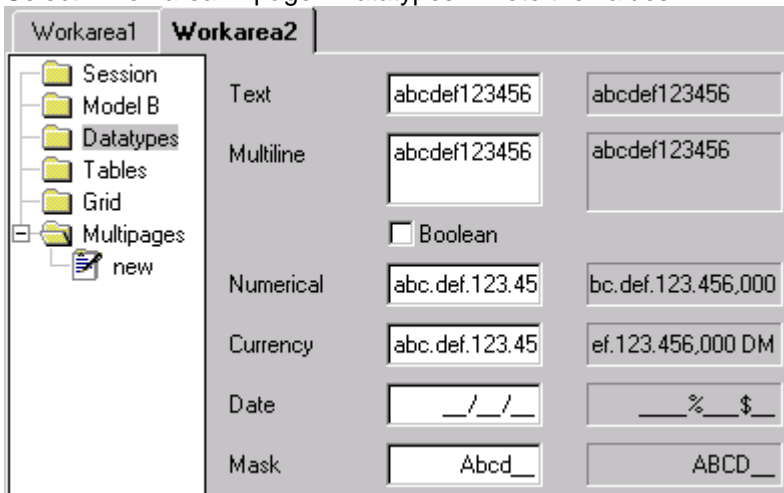
The layout can have a single start attribute. The value of the start attribute can be specified in the layout or the model.

Do the following:

1. If the layout created in T39 is not open in the Designer: Open 39.vpl.
2. Set application property <Start attribute> = <Constant>.
3. Set <Name> = <a_3>.
4. Set <Value> = <abcdef123456>.



5. Save the layout as 40.vpl.
6. Test the layout.
7. Select <Workarea2> page <Datatypes>. Note the values:

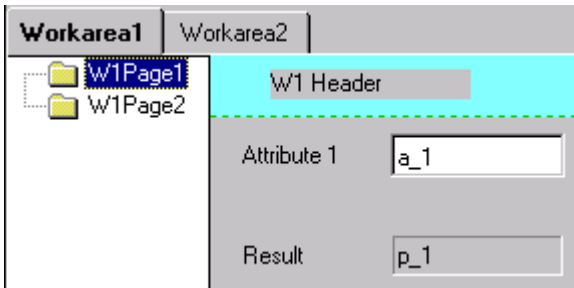


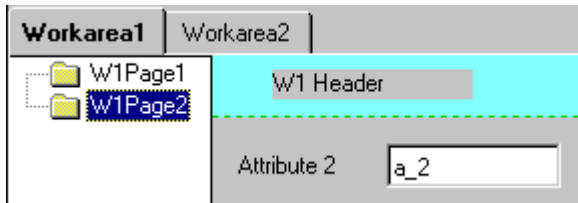
T41. Parameter search

When an attribute value is entered in the runtime layout, the focus can automatically be transferred to the layout element for a second attribute that is required along with the first attribute for a computation. Details.

Do the following:

1. If the layout created in T40 is not open in the Designer: Open 40.vpl.
2. Move the <Attribute 2> Label and EditField from <Workarea1> <W1Page1> to <W1Page2>.





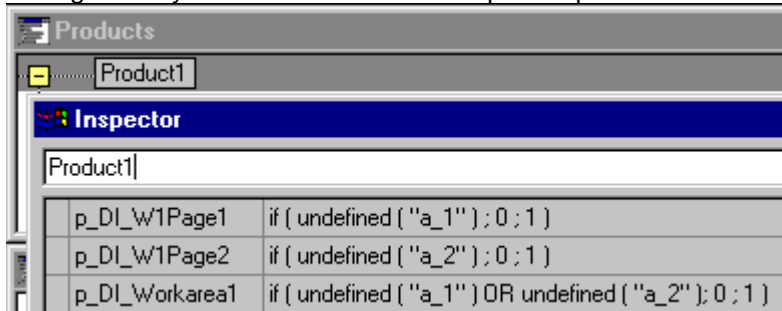
3. Save the layout as 41.vpl.
4. Test the layout.
5. For <Attribute 1>: Enter 123.
6. Click on the <Workarea2> tab. Note that <Workarea1> <W1Page2> is displayed (not <Workarea2>), since <Attribute 2> is required for computing Result <p_1>.
7. Enter 456 for <Attribute 2>.
8. Click on <Workarea2> tab. Note that <Workarea2> is displayed (the required attributes for Result have been entered).

T42. Data indicator

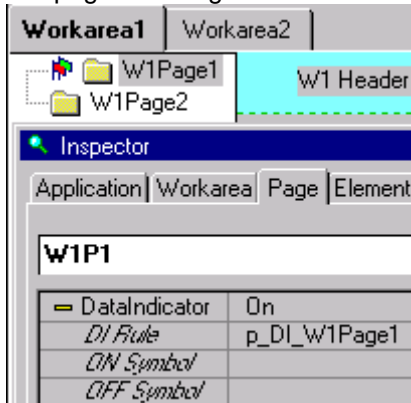
A symbol can be used to indicated if all required data on a workarea or page has been entered. Details.

Do the following:

1. If the layout created in T41 is not open in the Designer: Open 41.vpl.
2. Change the layout Product model to 41.vpm. 41.pms contains the DI (data indicator) rules:

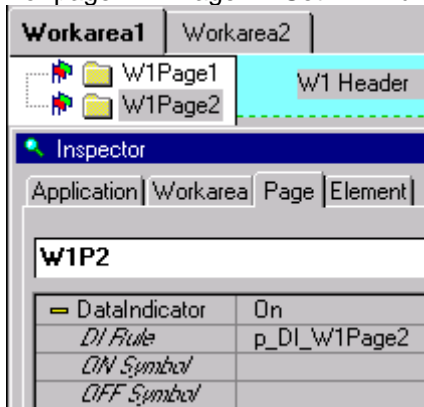


3. For page <W1Page1>: Set <DataIndicator> = <On>. Note the design-time layout data indicator symbol for the page icon in the frame.
4. For page <W1Page1>: Set <DI Rule> = <p_DI_W1Page1>.

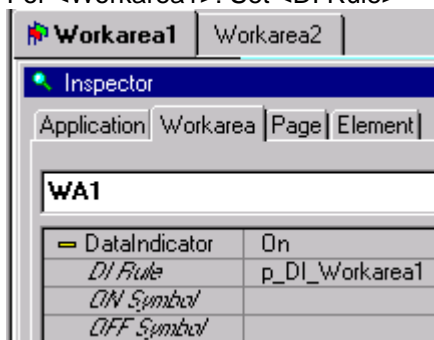


5. For page <W1Page2>: Set <DataIndicator> = <On>.

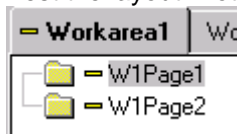
- For page <W1Page2>: Set <DI Rule> = <p_DI_W1Page2>.



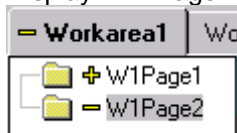
- For <Workarea1>: Set <DataIndicator> = <On>.
- For <Workarea1>: Set <DI Rule> = <p_DI_Workarea1>.



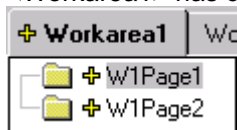
- Save the layout as 42.vpl.
- Test the layout. Note the "-" icon for the workarea and page tabs.



- For <Attribute 1>: Enter 123.
- Display <W1Page2>. Note that the symbol for <W1Page1> has changed to a <+>.



- For <Attribute 2>: Enter 456.
- Display <W1Page1> (to change the focus). Note that the symbol for <W1Page2> and <Workarea1> has changed to a <+>:



Recomputing result

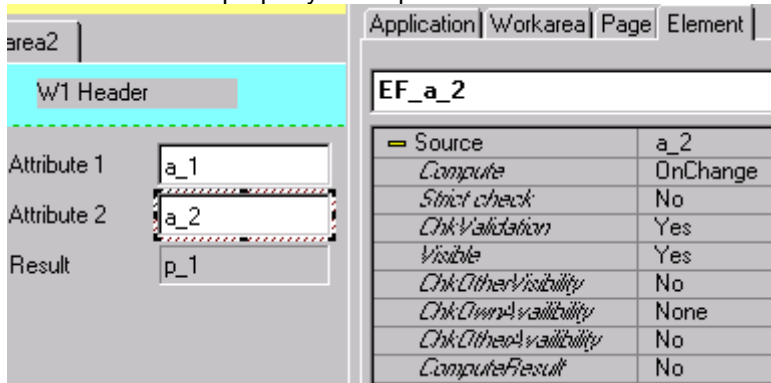
This part demonstrates how the layout can control how the data entered in the layout is processed by the model logic and the results displayed in the layout.

T43. Calculate: Page change

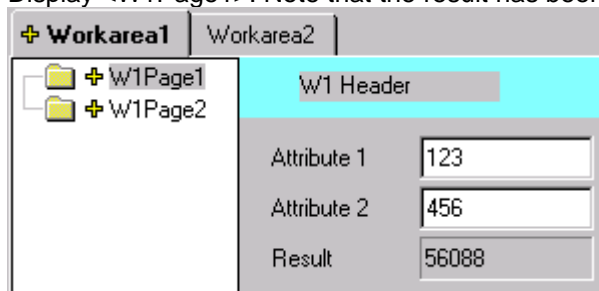
If [application property <Calculation> = <Automatic>]
 AND [entry element property <Compute result> = <No>]

AND [the value entered in the element is entered or modified]
 AND [all other values for a result have already been entered] :
 The result is recalculated when a different page is selected.
 Do the following:

1. If the layout created in T42 is not open in the Designer: Open 42.vpl.
2. Move <Attribute 2> Label and EditField to <Workarea1>.
3. Set <Attribute 2> property <ComputeResult> = <No>.



4. Save the layout as 43.vpl.
5. Test the layout.
6. Enter for <Attribute 1>: 123.
7. Enter for <Attribute 2>: 456. Note that the result is not calculated.
8. Click in the EditField for <Attribute 1> (to change the focus). Note that the result is not computed.
9. Display <W1Page2>.
10. Display <W1Page1>. Note that the result has been calculated.



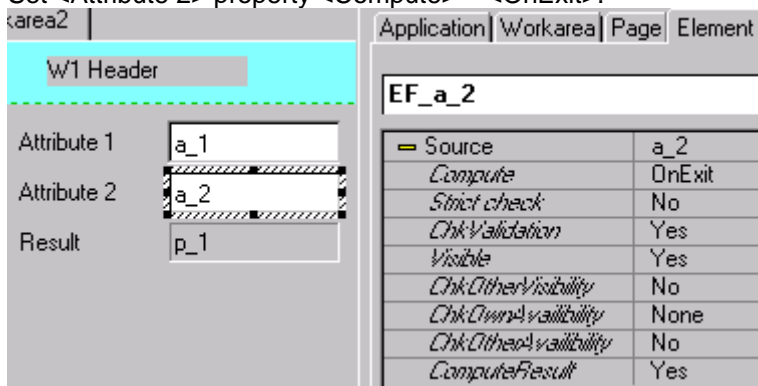
T44. Calculate: Focus change

If [application property <Calculation> = <Automatic>]
 AND [entry element property <Compute result> = <Yes>]
 AND [entry element property <Compute> = <OnExit>] :
 AND [the value entered in the element is entered or modified]
 AND [all other values for a result have already been entered] :
 The result is recalculated when the focus changes.

Do the following:

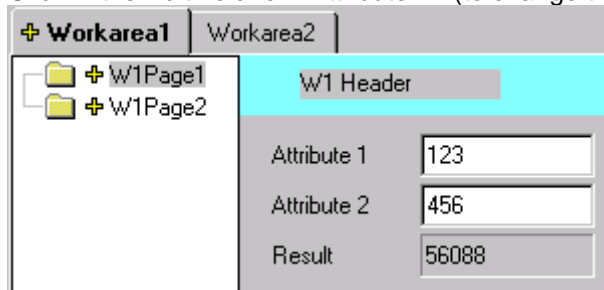
1. If the layout created in T43 is not open in the Designer: Open 43.vpl.
2. Set <Attribute 2> property <ComputeResult> = <Yes>.

- Set <Attribute 2> property <Compute> = <OnExit>.



Application	Workarea	Page	Element
EF_a_2			
Source		a_2	
Compute		OnExit	
Strict check		No	
ChkValidation		Yes	
Visible		Yes	
ChkOtherVisibility		No	
ChkOwnAvailability		None	
ChkOtherAvailability		No	
ComputeResult		Yes	

- Save the layout as 44.vpl.
- Test the layout.
- Enter for <Attribute 1>: 123.
- Enter for <Attribute 2>: 456. Note that the result is not calculated.
- Click in the EditField for <Attribute 1> (to change the focus). Note that the result is computed.



Workarea1 | Workarea2

W1 Header

Attribute 1: 123

Attribute 2: 456

Result: 56088

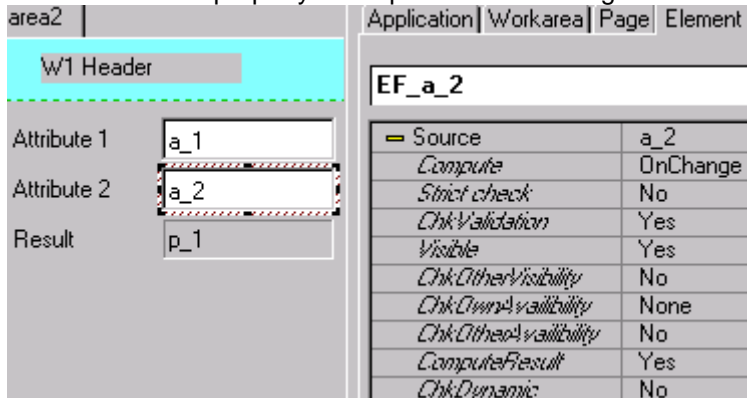
T45. Calculate: Data entry

If [application property <Calculation> = <Automatic>]
 AND [entry element property <Compute result> = <Yes>]
 AND [entry element property <Compute> = <OnChange>]
 AND [the value entered in the element is entered or modified]
 AND [all other values for a result have already been entered] :

The result is recalculated immediately.

Do the following:

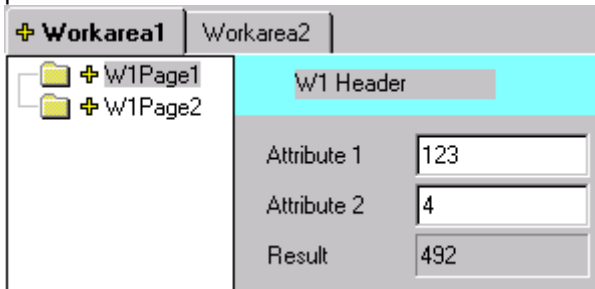
- If the layout created in T44 is not open in the Designer: Open 44.vpl.
- Set <Attribute 2> property <Compute> = <OnChange>.



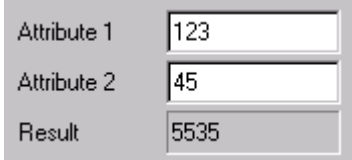
Application	Workarea	Page	Element
EF_a_2			
Source		a_2	
Compute		OnChange	
Strict check		No	
ChkValidation		Yes	
Visible		Yes	
ChkOtherVisibility		No	
ChkOwnAvailability		None	
ChkOtherAvailability		No	
ComputeResult		Yes	
ChkDynamic		No	

- Save the layout as 45.vpl.
- Test the layout.
- Enter for <Attribute 1>: 123.

6. Enter for <Attribute 2>: 4. Note that the result is calculated immediately after the 4 key is pressed.



7. Press the 5 key. Note that the result is again immediately calculated.



T46. Calculate: Pushbutton click

If [application property <Calculation> = <Manual>

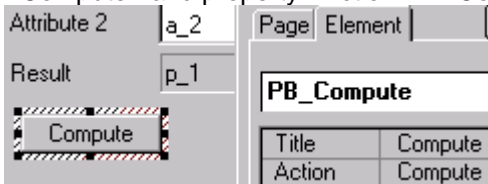
AND [the value entered in the element is entered or modified]

AND [all other values for a result have already been entered] :

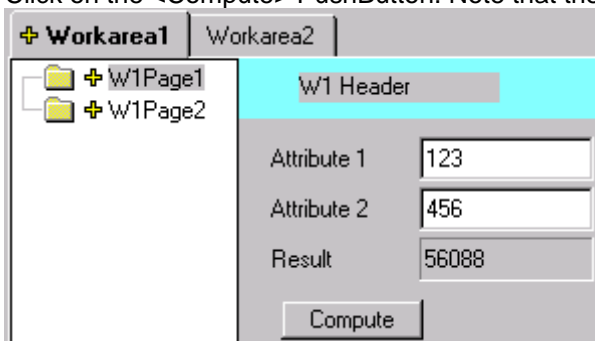
The result is recalculated only when a PushButton with <Action> = <Compute> is clicked.

Do the following:

1. If the layout created in T45 is not open in the Designer: Open 45.vpl.
2. Set application property <Calculation> = <Manual>.
3. In <Workarea1> <W1Page1>: Add a PushButton with <Name> = <PB_Compute>, <Title> = <Compute> and property <Action> = <Compute>.



4. Save the layout as 46.vpl.
5. Test the layout.
6. Enter for <Attribute 1>: 123.
7. Enter for <Attribute 2>: 456. Note that the result is not calculated.
8. Click to change the focus or page. Note that the result is not calculated.
9. Click on the <Compute> PushButton. Note that the result is calculated.



Note: A different page may be displayed when clicking the <Compute> Pushbutton, since data is required for other results.

Visibility

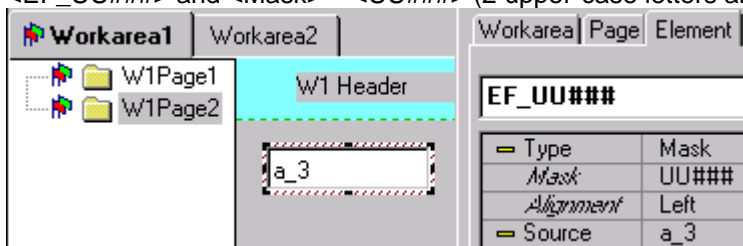
The visibility of elements can be set during design-time (sometimes useful when creating the layout) and run-time (for example when certain components are visible only under certain conditions).

T47. Visibility: Design-time

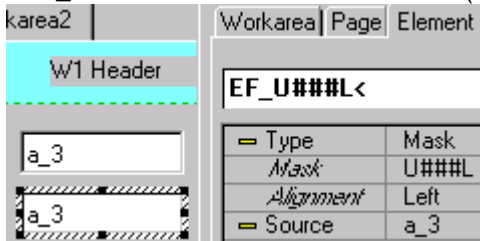
An element can be hidden in the layout during design time. You will encounter an example layout in which this is useful later in this tutorial.

Do the following.

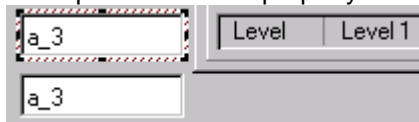
1. If the layout created in T46 is not open in the Designer: Open 46.vpl.
2. In <Workarea1> page <W1Page2>: Add a MaskField with <Source> = <a_3>, <Name> = <EF_UU###> and <Mask> = <UU###> (2 upper-case letters and 3 digits).



3. In <Workarea1> page <W1Page2>: Add a MaskField with <Source> = <a_3>, <Name> = <EF_U###L> and <Mask> = <U###L> (upper-case letter + 3 digits + lower-case letter).



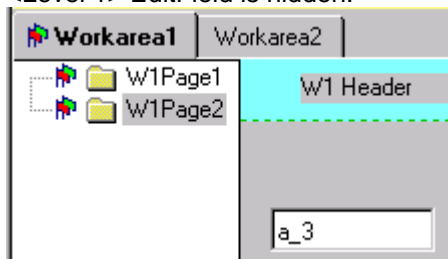
4. For top EditField: Set property <Level> = <Level 1>.



5. For lower EditField: Set property <Level>=<Level 2>.



6. Save the layout as 47.vpl.
7. From the main menu: Select <Options> / <Levels> / <Level 1> (uncheck). Note that the <Level 1> EditField is hidden:



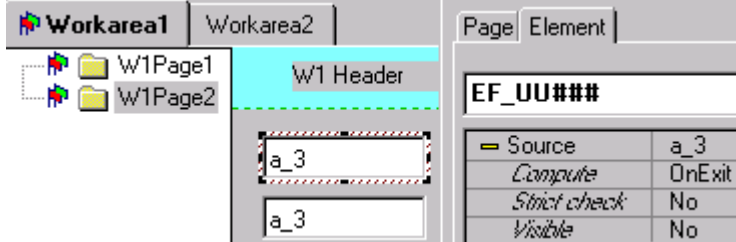
8. From the main menu: Select <Options> / <Levels> / <Level 2> (uncheck). Note that the <Level 2> EditField is hidden.
9. From the main menu: Select <Options> / <Levels> / <All levels>. Note that the EditFields are again visible.

T48. Visibility: Run-time static

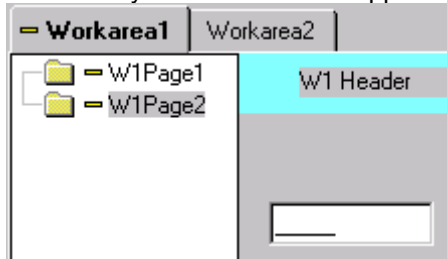
An element can be hidden in the runtime layout. In certain cases an element that should not be displayed is still required in the layout.

Do the following.

1. If the layout created in T47 is not open in the Designer: Open 47.vpl.
2. In <Workarea1> page <W1Page2>: For the upper EditField: Set property <Visible> = <No>.



3. Save the layout as 48.vpl.
4. Test the layout. Note that the upper EditField is not visible.

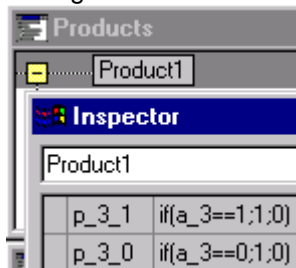


T49. Visibility: Run-time dynamic

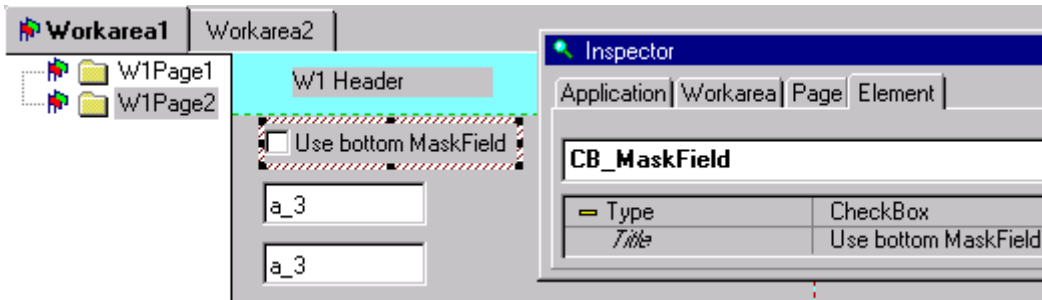
An element in a layout can be hidden or displayed based on a run-time condition in the model or run-time input in the layout (without any processing in the model). This page demonstrates the most common and simple method for implementing run-time dynamic visibility. More run-time dynamic details.

Do the following:

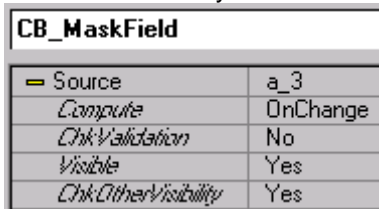
1. If the layout created in T48 is not open in the Designer: Open 48.vpl.
2. Change the Product model to 49.vpm. 49.pms contains the following:



- In <Workarea1> page <W1Page2>: Add a PushButton with <Name> = <CB_MaskField> and <Title> = <Use bottom MaskField>.



- For the PushButton: Set <Source> = <a_3>, <Compute> = <OnChange> and <ChkOtherVisibility> = <Yes>:



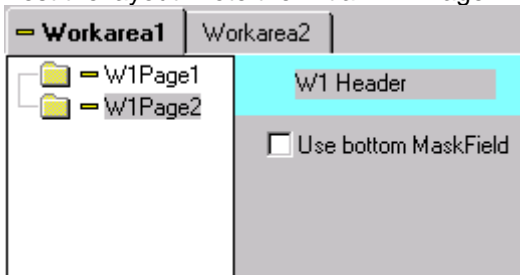
- For the upper EditField: Set <Source> subproperty <Visible> = <Value> and <Property> = <p_3_0>:



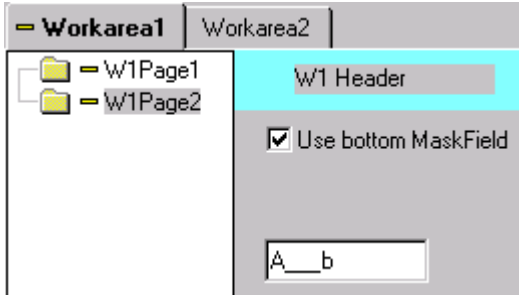
- For the lower EditField: Set <Source> subproperty <Visible> = <Value> and <Property> = <p_3_1>:



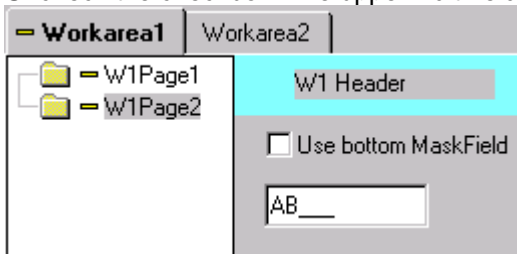
- Save the layout as 49.vpl.
- Test the layout. Note the initial <W1Page2>:



- Check the checkbox. The lower EditField is displayed (the upper is hidden):



10. Uncheck the checkbox. The upper EditField is displayed (the lower is hidden):



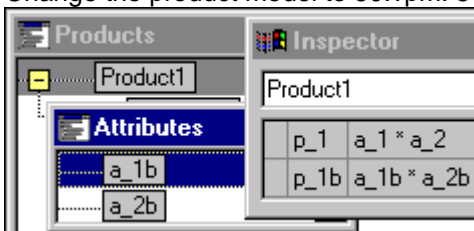
Availability

Sometimes an element should not be available in the layout, but should still be visible. This part demonstrates how to implement this with the Designer.

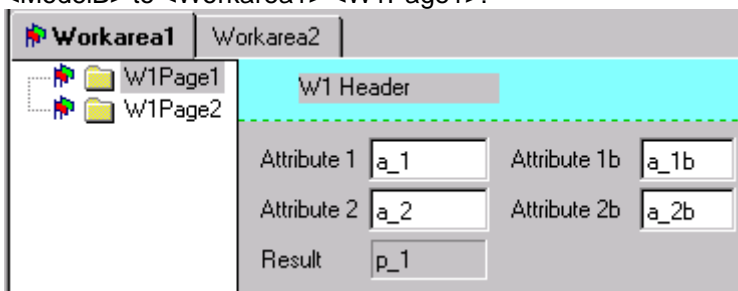
T50. Availability: Static

The availability of an element can be specified as always being not available. Do the following:

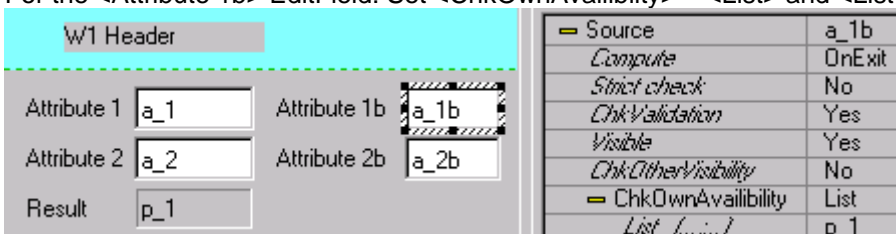
1. If the layout created in T49 is not open in the Designer: Open 49.vpl.
2. Change the product model to 50.vpm. 50.pms contains the following:



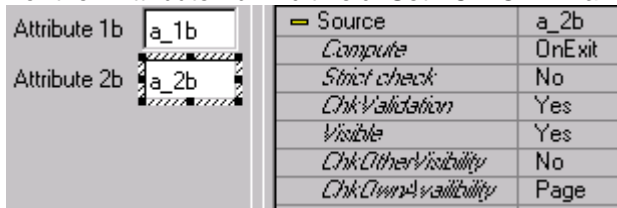
3. Copy the Label's and EditField's for <Attribute 1> and <Attribute2> in <Workarea2> page <ModelB> to <Workarea1> <W1Page1>:



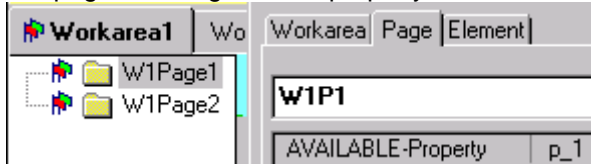
4. Rename the copied elements to avoid warning messages.
5. For the <Attribute 1b> EditField: Set <ChkOwnAvailability> = <List> and <List (...;...)> = <p_1>.



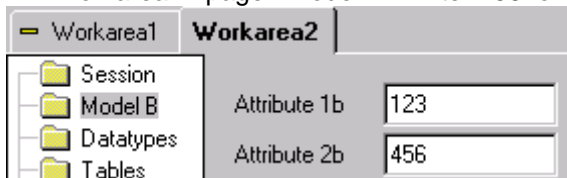
- For the <Attribute 2b> EditField: Set <ChkOwnAvailability> = <Page>.



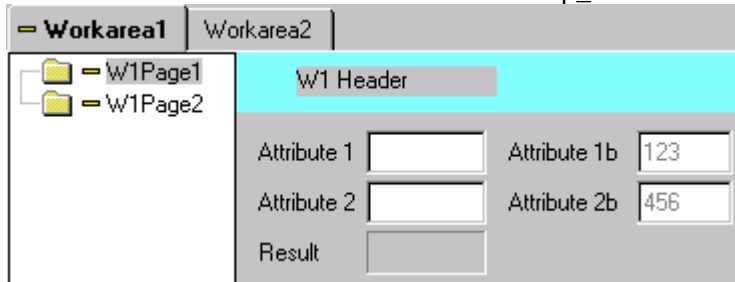
- For page <W1Page1>: Set property <AVAILABLE-Property> = <p_1>.



- Save the layout as 50.vpl.
- Test the layout.
- In <Workarea2> page <ModelB>: Enter 123 for <Attribute 1b>.
- In <Workarea2> page <ModelB>: Enter 456 for <Attribute 2b>.



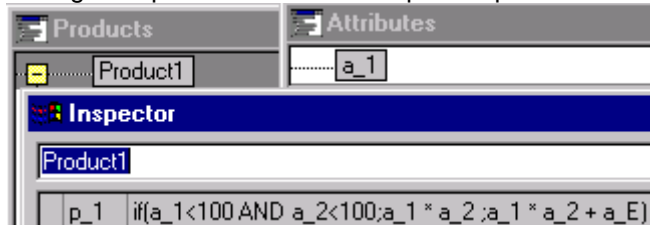
- In <Workarea1> page <W1Page1>: Note that the <Attribute 1b> EditField is not available since the List specifies <p_1b>, which has no element on the page. Note that <Attribute 2b> EditField is not available since no element for <p_1b> exists on the page.



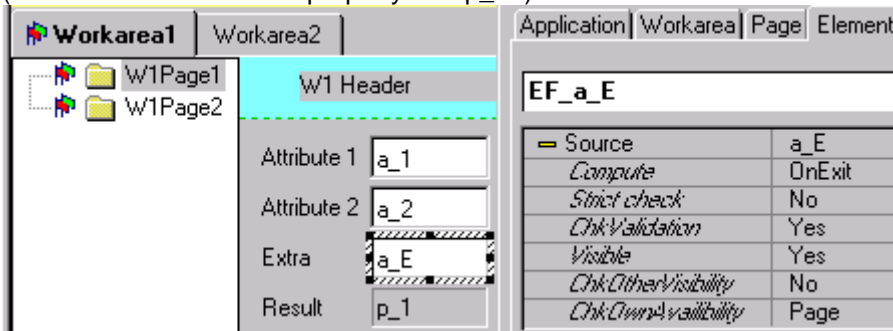
T51. Availability: Dynamic

The availability of an element can be dynamic in the runtime layout.
Do the following:

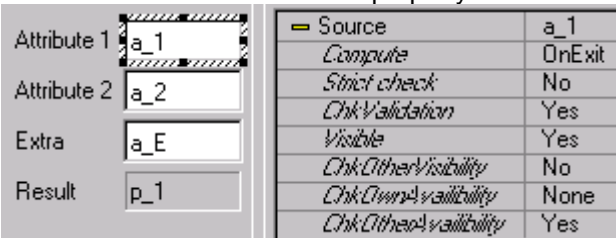
- If the layout created in T50 is not open in the Designer: Open 50.vpl.
- Change the product model to 51.vpm. 51.pms contains the following:



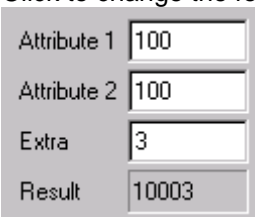
3. Add a Label and EditField with <Source> = <a_E> and <ChkOwnAvailability> = <Page> (remember <AVAILABLE-property> = <p_1>):



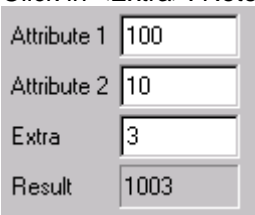
4. For <Attribute 1> EditField: Set property <ChkOtherAvailability> = <Yes>:



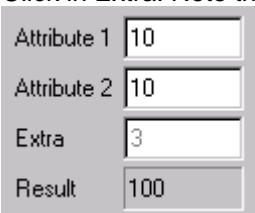
5. Save the layout as 51.vpl.
6. Test the layout. Note that the Extra EditField is not available.
7. For <Attribute 2>: Enter 100. Note that the Extra EditField is not available.
8. For <Attribute 1>: Enter 100. Note that the Extra EditField is not available.
9. Click in <Attribute 2> EditField. The focus change caused other availability to be checked. Data can now be entered in the <Extra> EditField.
10. Enter 3 in the <Extra> EditField.
11. Click to change the focus. Note the result:



12. Change <Attribute 2> to 10.
13. Click in <Extra>. Note that it is still available.



14. Change <Attribute 1> to 10.
15. Click in Extra. Note that it is not available.



Where to go from here

At this point you have a basic understanding of Designer functionality.

Now it is recommended to do the following:

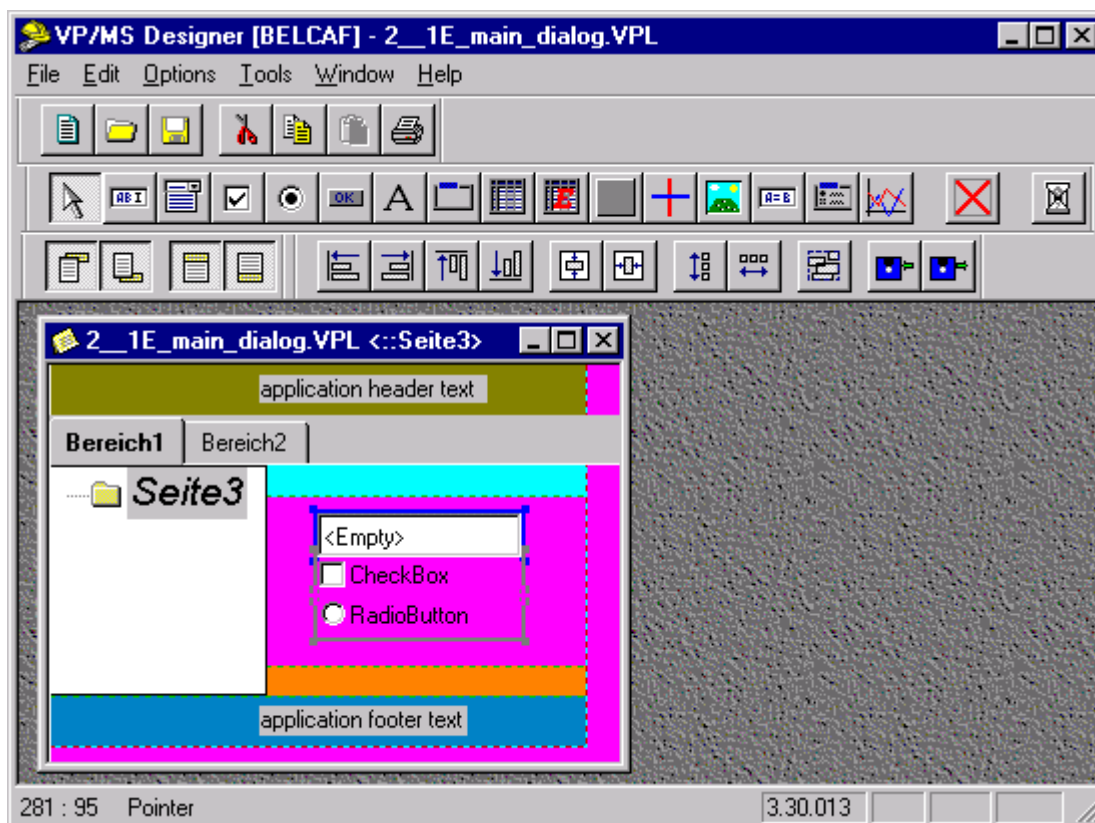
1. Read in detail Designer concepts.
2. Study the Designer properties.
3. Review how to trouble-shoot common problems and the answers to frequently-asked questions.
4. Analyze the example models.

Tools and dialogs

The following is a list of the tools and dialogs available within Designer.

- Main dialog
- Options dialog
- Inspector
- Test
- Find/Find Next
- Elements reordering dialog
- Autosize
- Synchronisierungs-Tool
- Synchronisations-Report
- Data elements
- CRF Preparation (Print)
- Screen shot

Main dialog

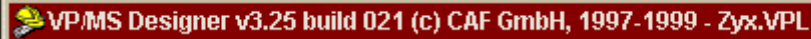


Title bar

The title bar contains the following information:

- The Designer version and build.

- The layout currently selected.

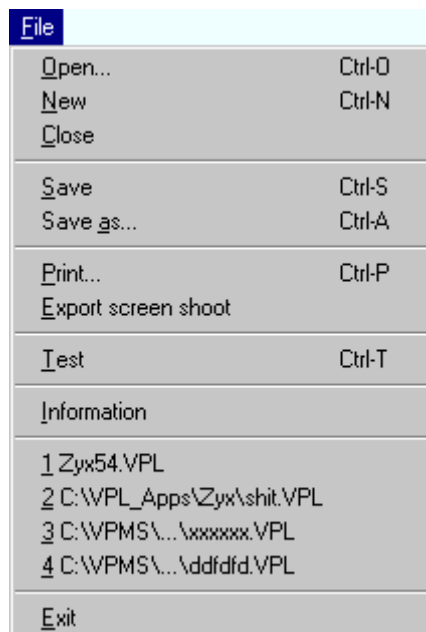


Menu bar

The complete menu bar as shown below appears when a layout has been opened in Designer.

Datei Bearbeiten Optionen Iools Fenster Hilfe

Menu: File



Menu: File / Open

Brings up the Windows Open dialog for selecting a .vpl file.

Toolbar icon.

Menu: File / New

Creates a new .vpl file and opens in the Designer.

Toolbar icon.

Menu: File / Close

Closes the .vpl file selected in the Designer.

A prompt for saving the file will appear if the file has unsaved changes.

Does not close the Designer.

The file can also be closed by clicking the Close button in the upper-right corner of the .vpl file dialog.

Toolbar icon.

Menu: File / Save

Saves the currently selected layout to a .vpl file.

Note: The .vpl file is compiled to a .vpc file during the save.

Same as File / Save as... if the selected .vpl file is new and has not been saved yet.

Toolbar icon.

Menu: File / Save as...

Brings up the Windows "Save as..." dialog. The layout is saved to the specified .vpl file.

Note: The .vpl file is compiled to a .vpc file during the save.

Menu: File / Print

Brings up the CRF preparation dialog for printing the layout.

Note: Has no effect if the layout contains no elements.

Toolbar icon.

Menu: File / Export screen shot

Brings up the dialog Select page for screen shot.

See saving workareas/pages to .bmp files for details.

Menu: File / Test

Tests the layout file:

If the layout has not been saved: Prompt for saving the layout appears (the .vpl file is compiled to a .vpc file during the save).

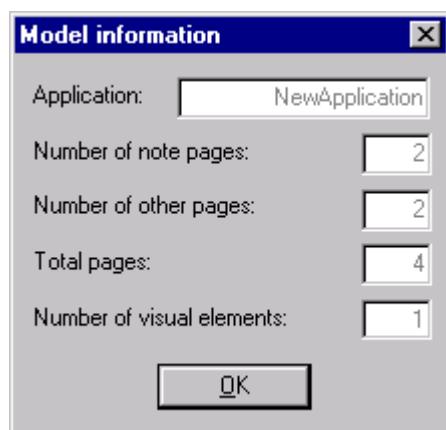
The compiled layout (.vpc) is opened in the Test tool.

Toolbar icon.

Menu: File / Information

Displays the following information about the currently selected model:

- The application name.
- Number of note pages.
- Number of other pages.
- Total number of pages.
- Number of visual elements.



Menu: File / (list of most recent files)

A list of the most recently opened .vpl layout files is included in the File menu. Selecting an entry will open that file in the Designer.

Example

The following File menu contains links to files.

<u>O</u> pen...	Ctrl-O
<u>N</u> ew	Ctrl-N
<u>C</u> lose	
<u>S</u> ave	Ctrl-S
Save <u>a</u> s...	Ctrl-A
<u>P</u> rint...	Ctrl-P
<u>T</u> est	Ctrl-T
<u>I</u> nformation	
<u>1</u> Zy05.VPL	
<u>2</u> Zy.VPL	
<u>E</u> xit	

Menu: File / Exit

Closes Designer.

Note: If a layout in the Designer has unsaved changes: A prompt will appear to save the changes.

Menu Edit

<u>E</u> dit	
Undo delete element	
<u>F</u> ind	Alt-F3
Find <u>N</u> ext	F3
<u>C</u> opy	Ctrl-C
<u>P</u> aste	Ctrl-V
<u>C</u> ut	Ctrl-X
<u>D</u> elete	
<u>S</u> elect All	
<u>G</u> roup	▶
Elements <u>r</u> eordering	

Menu: Edit / Undo

Causes the last action to be undone.

The name of this menu item sometimes describes what type of action will be undone.

Example

If an element is added to a layout, then the menu selection will be named "Edit / Undo add element".

Menu: Edit / Copy

Select to:

- Copy selected text
- Copy a selected element
- Copy a selected group of elements to the clipboard.

Note: Pressing Ctrl-C has the same effect.

Menu: Edit / Paste

Paste the contents of the clipboard (text, element, or group of elements) from the clipboard to the layout.

Note: Pressing Ctrl-V has the same effect.

Menu: Edit / Cut

Copies the selected text, element, or group of elements to the clipboard and then deletes the selected items.

Select to:

- Cut selected text
- Cut a selected element
- Cut a selected group of elements

Note: Pressing Ctrl-X has the same effect.

Menu: Edit / Delete

Deletes the selected text, element, or group of elements from the layout.

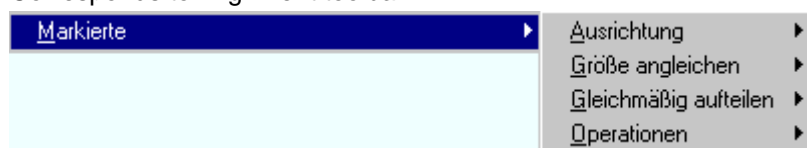
Note: Pressing the Del key has the same effect for only text or an element (not a group of elements).

Menu: Edit / Select all

Selects all elements on the displayed workarea page.

Menu: Edit / Group

The Edit / Group submenu provides operations for groups of elements. Corresponds to Alignment toolbar.



Menu: Edit / Group / Group Align

A group of elements can be aligned.



Menu: Edit / Group /Alignment / Align Left

Select to align the left edge of the group elements.
Same as clicking the icon Align left.

Menu: Edit / Group /Alignment /Aline Right

Select to align the right edge of the group elements.
Same as clicking the icon Align right.

Menu: Edit / Group /Alignment / Aline Up

Select to align the top edge of the group elements.
Same as clicking the icon Align up.

Menu: Edit / Group /Alignment / Aline Down

Select to align the bottom edge of the group elements.
Same as clicking the icon Align down.

Menu: Edit / Group / Make same size

All the elements selected in a group can be set to the same height and/or width.



Menu: Edit / Group / Make same size/ Align Height

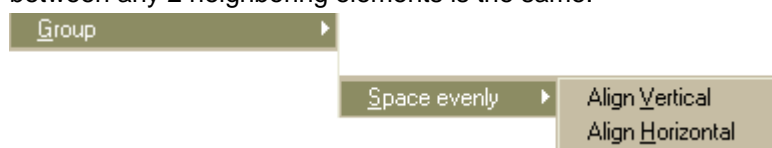
Changes the height of all selected elements to the height of the last element selected.

Menu: Edit / Group / Make same size/ Align Width

Changes the width of all selected elements to the height of the last element selected.

Edit / Group / Space Evenly

All of the elements selected in a group can be positioned so that the horizontal and/or vertical space between any 2 neighboring elements is the same.



Menu: Edit / Group / Space Evenly / Align Vertical

All elements in the group (with the exception of the 2 highest elements) are moved vertically such that:
[The vertical space between the centers of any 2 vertically neighboring elements] = [The vertical space between the centers of the highest and the second highest elements].

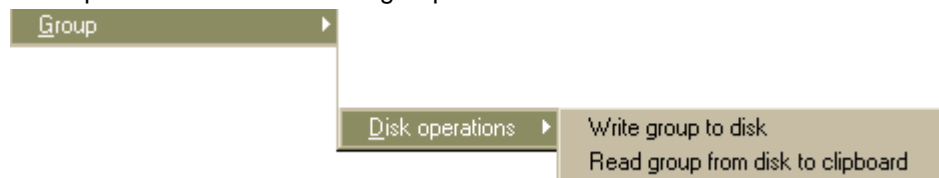
Menu: Edit / Group / Space Evenly / Align Horizontal

All elements in the group (with the exception of the 2 leftmost elements) are moved horizontally such that:

[The horizontal space between the centers of any 2 horizontally neighboring elements] = [The horizontal space between the centers of the leftmost and the second leftmost elements].

Edit / Group / Disk Operations

Disk operations allow an entire group of elements to be written or read to / from the clipboard.



Menu: Edit / Group / Disk Operations / Write Group to Disk

Copies the elements selected in the layout to a .vpg file.

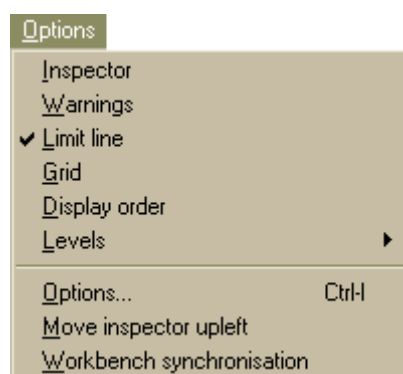
Menu: Edit / Group / Disk Operations / Read Group from disk to clipboard

Copies the contents of a .vpg file (a group of elements) to the clipboard.

Menu: Edit / Elements reordering

Displays the Elements (re)ordering dialog.

Menu: Options



Menu: Options / Inspector

Opens the Inspector.

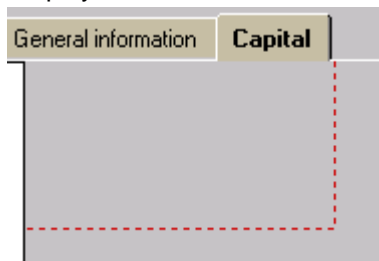
Menu: Options / Warnings

Opens or closes the Warning message window.



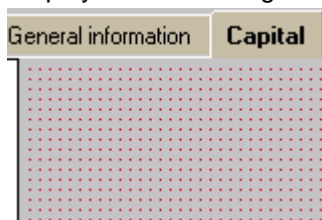
Menu: Options / Limit Line

Displays or hides the red dotted limit line in the layout.



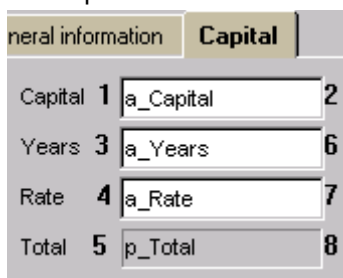
Menu: Options / Grid

Displays or hides the grid in the layout.



Menu: Options / Display Order

Displays or hides the order of the elements in the layout.
Concept.



Menu: Options / Levels

Displays or hides all elements on the selected level.



Menu: Options / Options...

Opens the Options dialog.

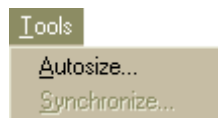
Menu: Options / Move Inspector Left

Moves the Inspector window to the upper left corner of the display.

Menu: Options / Workbench Synchronization

Synchronizes the layout with the model opened in the Workbench.

Menu: Tools



Menu: Tools / Autosize

Opens the Autosize dialog.

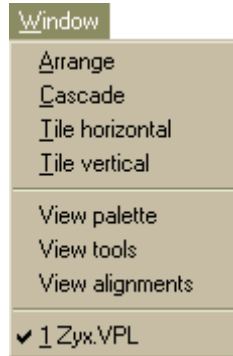
Menu: Tools / Synchronize

Opens the dialog "Synchronize".

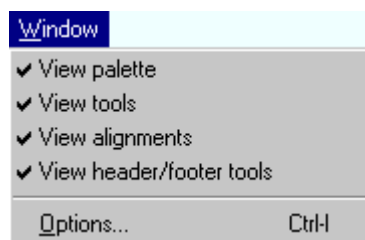
Menu: Window

Click on the diagram below for information about that window item.

Menu (1 or more layouts opened)



Menu (no layouts opened)

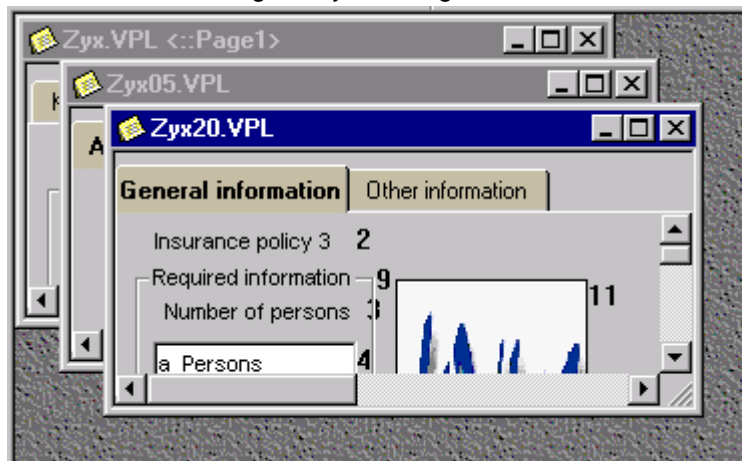


Menu: Window / Arrange

Arranges the windows in the Designer.

Menu: Window / Cascade

Arranges the layouts in the Designer in a cascade format. All layouts are changed to a single size that will fit within the Designer layout design area.

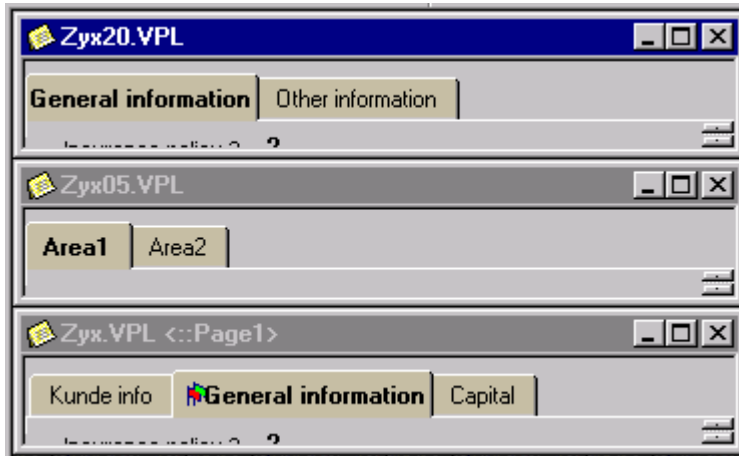


Menu: Window / Tile horizontal, vertical

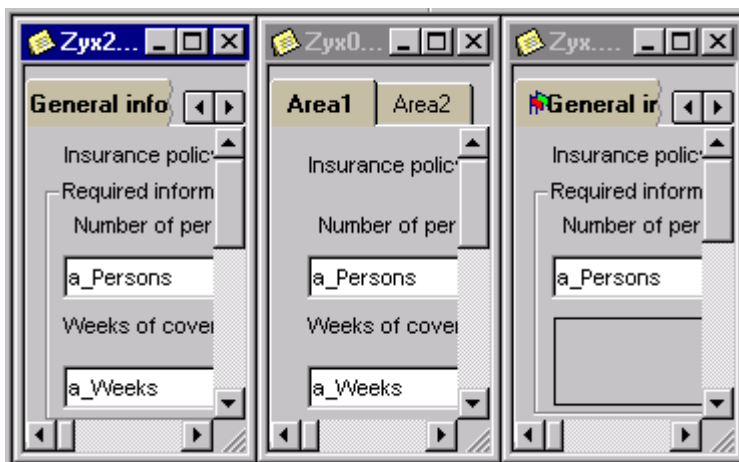
Arranges the layouts in the Designer in a tile format. All layouts are changed to a single size that will fit within the Designer layout design area.

Examples

Tile horizontal:



Tile vertical:



Menu: Window / View palette, View tools, View alignments

Displays or hides the following toolbars:

- Palette bar (common)
- Tools bar (visual palette)
- Alignment bar (groups)

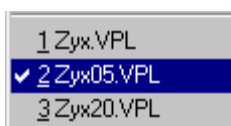
Menu: Window / (layout list)

A list of all layouts open in Designer.

The currently selected layout is marked with a check mark.

See also: List of open and previously opened files.

Example



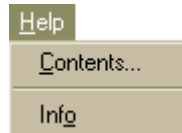
Menu: Window / View header/footer tools

Displays / hides the header/footer toolbar.

Menu: Window / Options

Opens the Options dialog.

Menu: Help



Menu: Help / Contents

Opens the Designer help file contents page.

Menu: Help / Info

Opens a window with:

- Information about Designer.
- Release information.
- Contact information

Toolbar: Tools (visual palette)



Toolbar icon: Pointer

Changes the cursor to a pointer for selecting elements (removes the crosshair images from the mouse pointer).

Toolbar icon: EditField

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds an EditField to the layout.

Toolbar icon: ComboBox

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds an ComboBox to the layout.

Toolbar icon: CheckBox

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds an CheckBox to the layout.

Toolbar icon: Radio button

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds an RadioButton to the layout.

Toolbar icon: Push Button

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds an PushButton to the layout.

Toolbar icon: Label

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a Label to the layout.

Toolbar icon: Border

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a Border to the layout.

Toolbar icon: Grid

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a Grid to the layout.

Toolbar icon: Extended Grid

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds an ExtendedGrid to the layout.

Toolbar icon: 3d Border

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a 3D Border to the layout.

Toolbar icon: Line

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a Line to the layout.

Toolbar icon: Multimedia element

Click to add crosshair images to the mouse pointer.

Clicking with the mouse cursor on a layout then adds a Multimedia element to the layout.

Toolbar icon: Result field

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds a ResultField to the layout.

Toolbar icon: RadioGroup

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds a RadioGroup to the layout.

Toolbar icon: Graphics element

Click to add crosshair images to the mouse pointer.
Clicking with the mouse cursor on a layout then adds a Graphics element to the layout.

Toolbar icon: Close layout file

Closes the .vpl file selected in the Designer.
A prompt for saving the file will appear if the file has unsaved changes.
Does not close the Designer.
The file can also be closed by clicking the Close button in the upper-right corner of the .vpl file dialog.

Toolbar icon: Test layout

Tests the layout file:
If the layout has not been saved: Prompt for saving the layout appears (the .vpl file is compiled to a .vpc file during the save).
The compiled layout (.vpc) is opened in the Test tool.

Toolbar: Palette (common)



Icon: New file

Creates a new .vpl file and opens in the Designer.

Icon: Open file

Brings up the Windows Open dialog for selecting a .vpl file.

Icon: Save file

Saves the currently selected layout to a .vpl file.
Note: The .vpl file is compiled to a .vpc file during the save.
Same as File / Save as... if the selected .vpl file is new and has not been saved yet.


 **Icon: Cut**

Copies the selected text, element, or group of elements to the clipboard and then deletes the selected items.

Select to:

- Cut selected text
- Cut a selected element
- Cut a selected group of elements

Note: Pressing Ctrl-X has the same effect.

 **Icon: Copy**

Click to:

- Copy selected text
- Copy a selected element
- Copy a selected group of elements

to the clipboard.

Note: Pressing Ctrl-C has the same effect.

 **Icon: Paste**

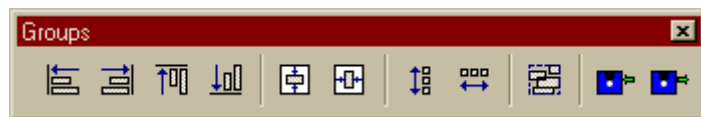
Paste the contents of the clipboard (text, element, or group of elements) from the clipboard to the layout.

Note: Pressing Ctrl-V has the same effect.

 **Icon: Print file**

Click to print (CRF preparation) the currently selected layout.

Toolbar: Alignment (groups)

 **Icon: Align Left**

Click to align the left edge of the group elements.

Same as selecting from the main menu Edit / Group / Alignment / Align left.

 **Icon: Align Right**

Click to align the left edge of the group elements.

Same as selecting from the main menu Edit / Group / Alignment / Align right.

 **Icon: Align Up**


Click to align the top edge of the group elements.

Same as selecting from the main menu Edit / Group / Alignment / Align up.


 **Icon: Align Down**

Click to align the left edge of the group elements.


Same as selecting from the main menu Edit / Group / Alignment / Align down.

 **Icon: Same height (Align height / Make same size vertical)**

Changes the height of all selected elements to the height of the last element selected.

 **Icon: Same width (Align width / Make same size horizontal)**


Changes the width of all selected elements to the width of the last element selected.

 **Icon: Space evenly vertical (Align vertical / Spread vertical evenly)**

All elements in the group (with the exception of the 2 highest elements) are moved vertically such that:

[The vertical space between the centers of any 2 vertically neighboring elements] =

[The vertical space between the centers of the highest and the second highest elements].

 **Icon: Space evenly horizontal (Align horizontal / Spread horizontal evenly)**

All elements in the group (with the exception of the 2 leftmost elements) are moved horizontally such that:

[The horizontal space between the centers of any 2 horizontally neighboring elements] =


[The horizontal space between the centers of the leftmost and the second leftmost elements].

 **Icon: Group all**

Selects all elements on the displayed workarea page.

 **Icon: Save group to disk**

Copies the elements selected in the layout to a .vpg file.

 **Icon: Read group from disk (and copy to clipboard)**

Copies the contents of a .vpg file (a group of elements) to the clipboard.

Toolbar: Header/footer tools



Icon: App header

Click to display / hide the application header.

Icon: App footer

Click to display / hide the application footer.

Icon: WArea header

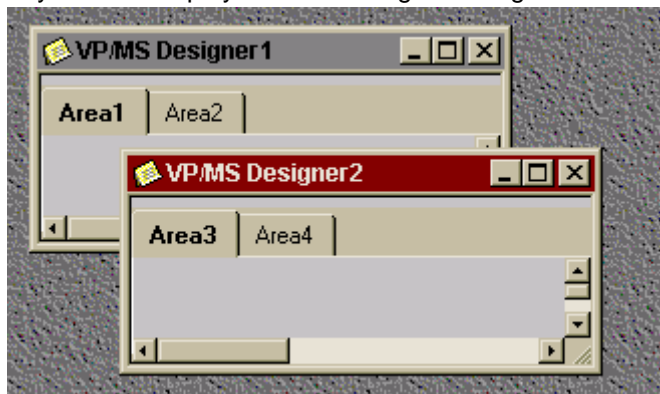
Click to display / hide the workarea header.

Icon: WArea footer

Click to display / hide the workarea footer.

Main dialog: Design area

Layouts are displayed in the Designer Design area.



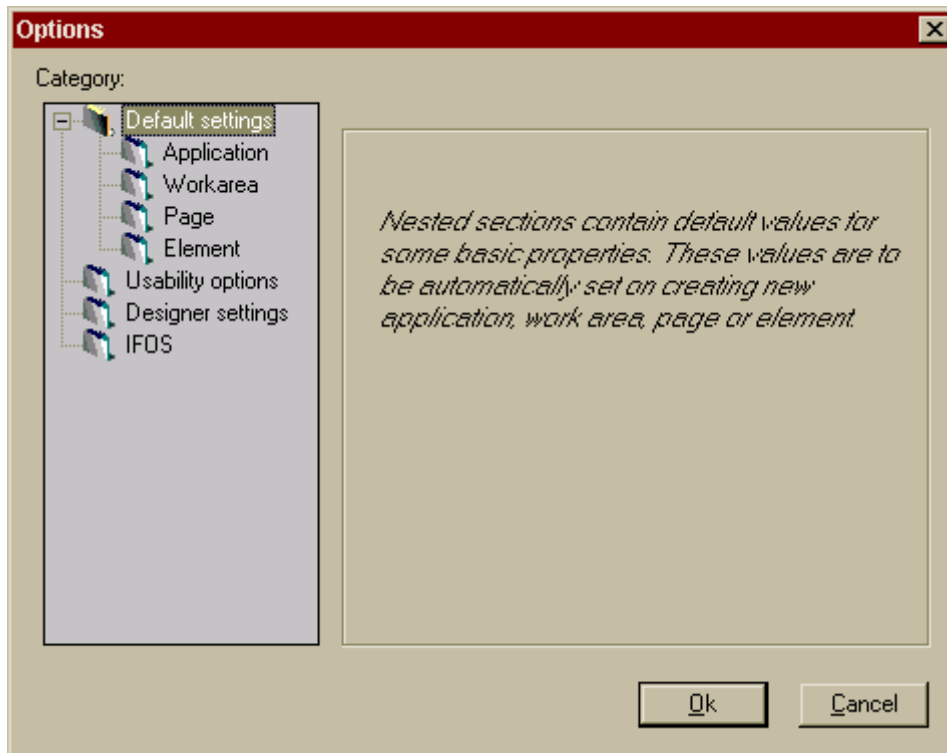
Main dialog: Status bar

The Status bar displays the following information:

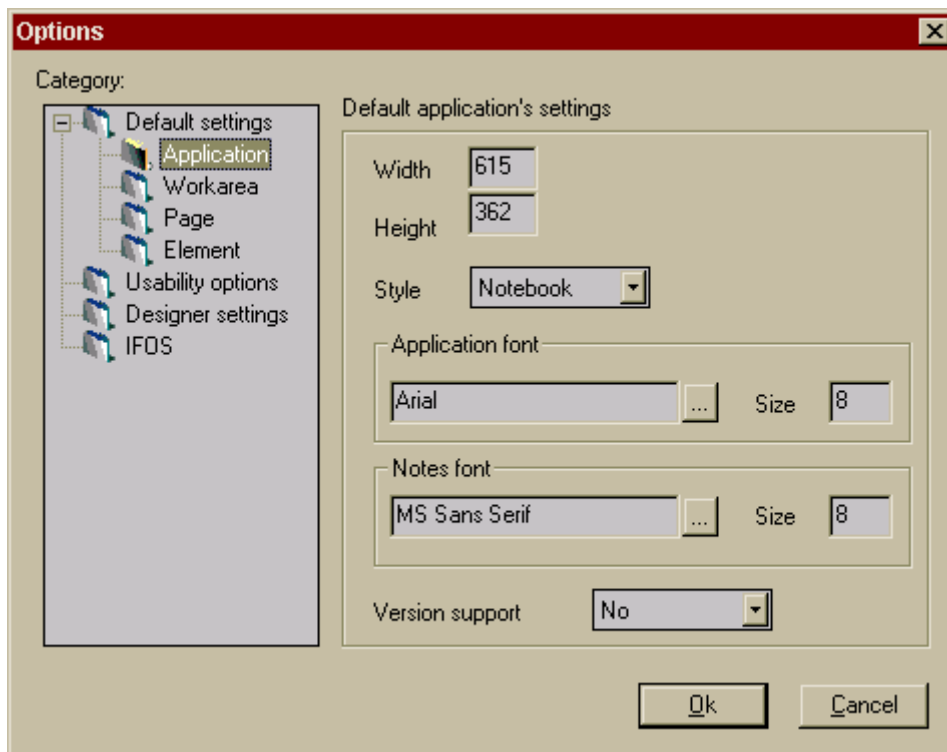
- Pointer location.
- Version number of Designer.
- Capital / Numeric / Scroll lock



Dialog: Options



Dialog: Options / Application

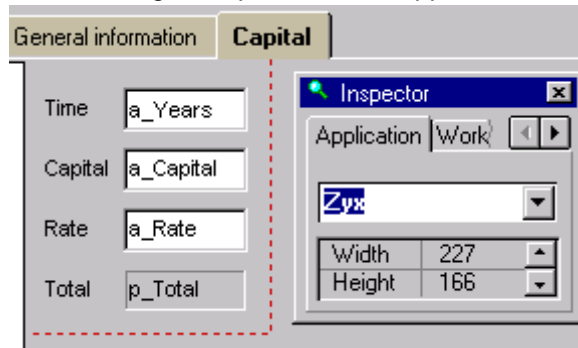


Dialog: Options / Application / Width/Height

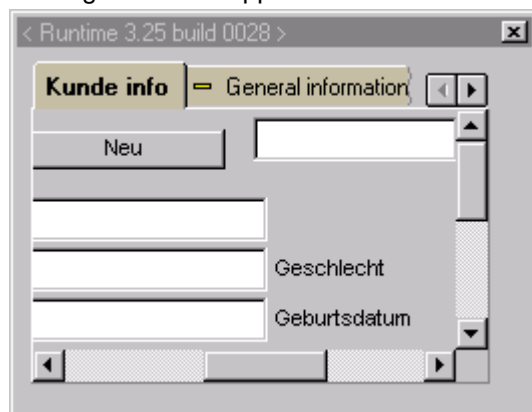
Sets the default width / height of an application (in pixels).

Example (of application width / height)

The following example shows an application with Width/Height set to 227/166.



Testing the above application would result in the following dialog.



Dialog: Options / Application / Style

Set the default application style property.

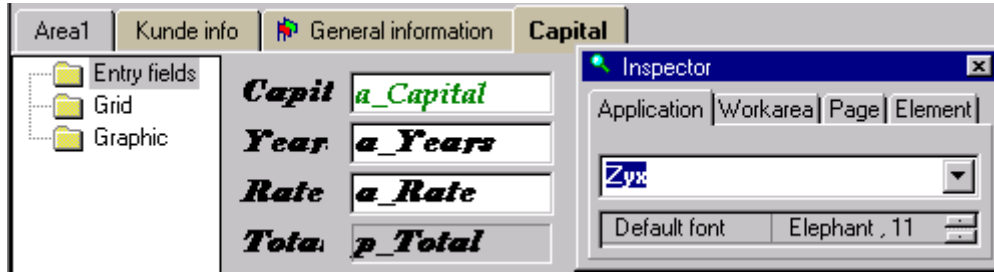
Dialog: Options / Application / Application, Notes (tab) fonts

Sets the default fonts for the entire application.

- The application font is the font for the contents of all pages, but NOT for frame text.
- The notes font is the font for the workarea tab text. NOTE: Changing the default value will only take effect after the layout has been reopened.

Example of application font setting

In the following example, the font for the application has been changed to Elephant,11. Note that the application font does not apply to the frame font.



Example of notes (tab) font setting

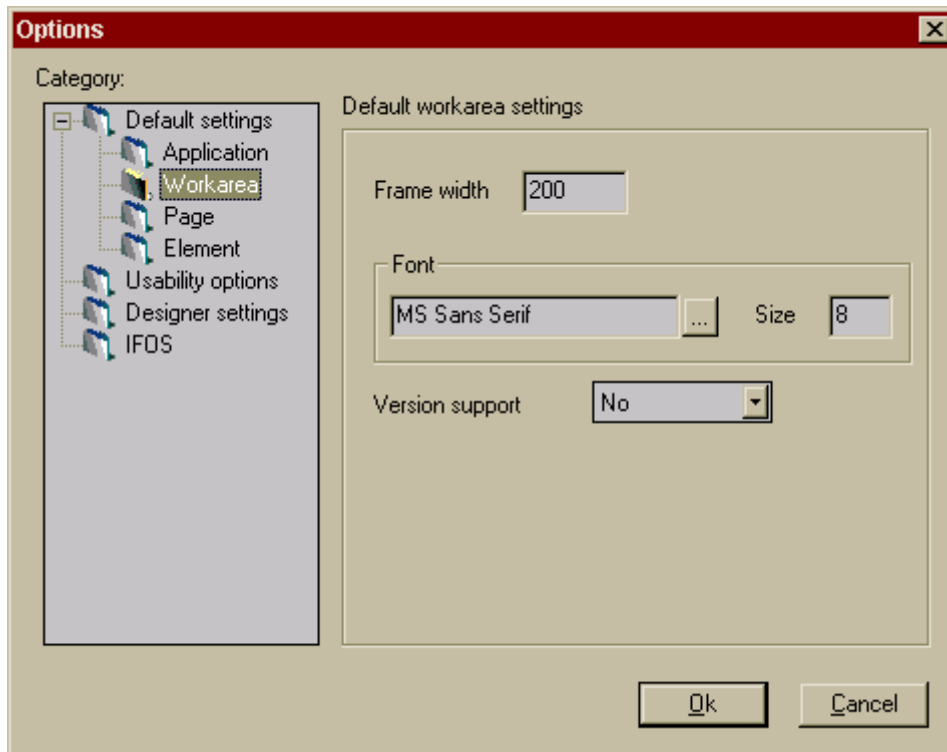
In the following example, the notes (tab) font was changed to Courier,12.



Dialog: Options / Application / Version Support

Specifies the default value for the version info property for applications.
Versioning in Designer.

Dialog: Options / Workarea

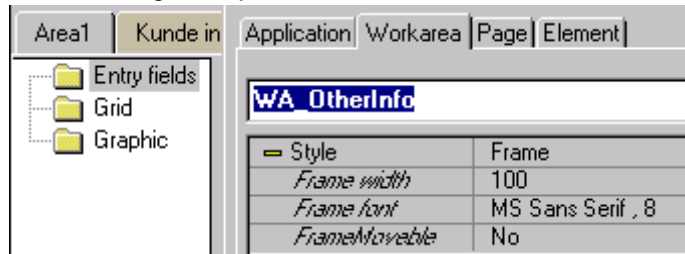


Dialog: Options / Workarea / Frame Width

Sets the default width of a frame in a workarea with style set to Frame.

Example

The following example shows a frame with width

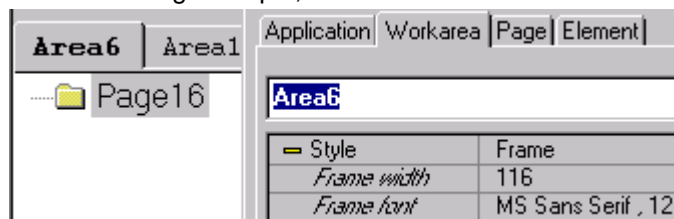


Dialog: Options / Workarea / (frame) font

Sets the default font for the frame for a workarea.

Example of workarea (frame) font setting

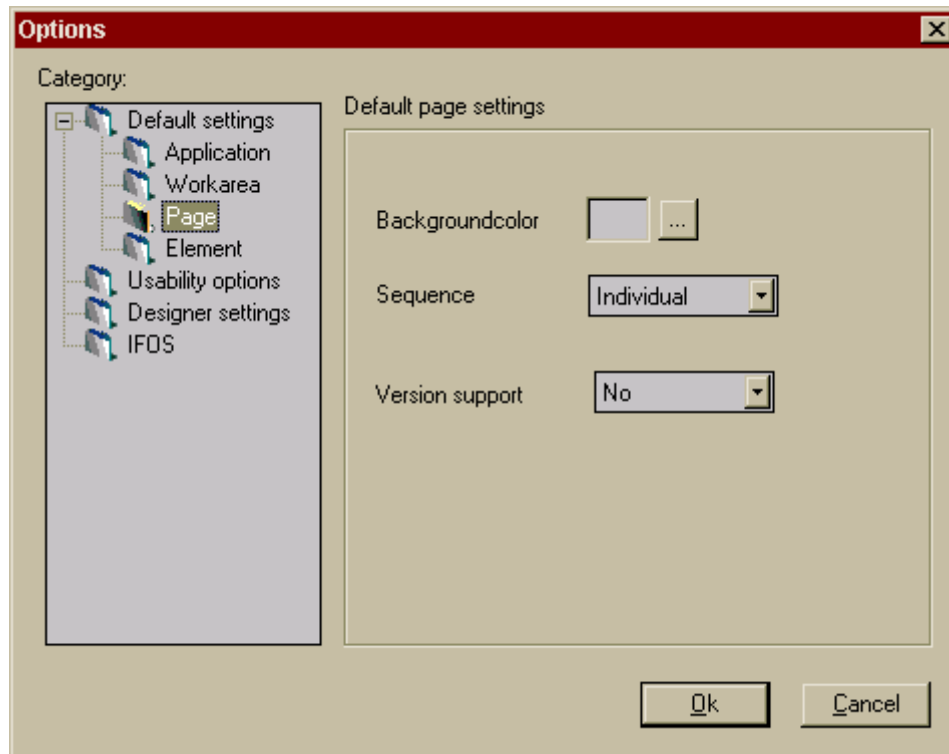
In the following example, the font for the frame has been changed to MS Sans Serif,12.



Dialog: Options / Workarea / Version Support

Specifies the default value for the version info property for workareas.

Dialog: Options / Page



Dialog: Options / Page / Background Color

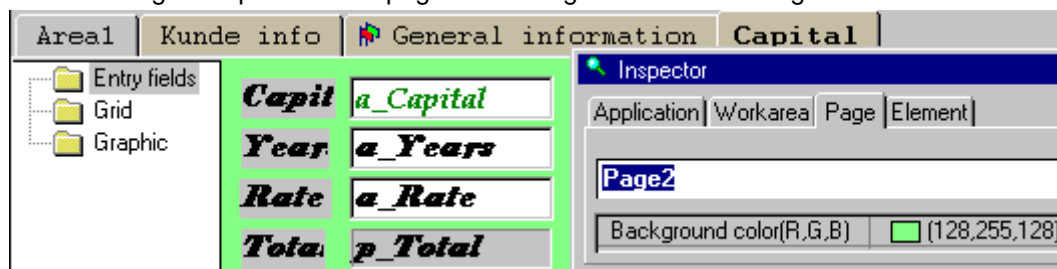
Sets the default Background color property of pages.

Notes:

- This has no effect on the frame and tab background color settings.
- The frame background cannot be changed.
- The tab background settings must be set individually (no default settings).
- Element background settings must be set individually (no default settings).

Example

The following example shows a page with background color set to green.



Dialog: Options / Page Sequence

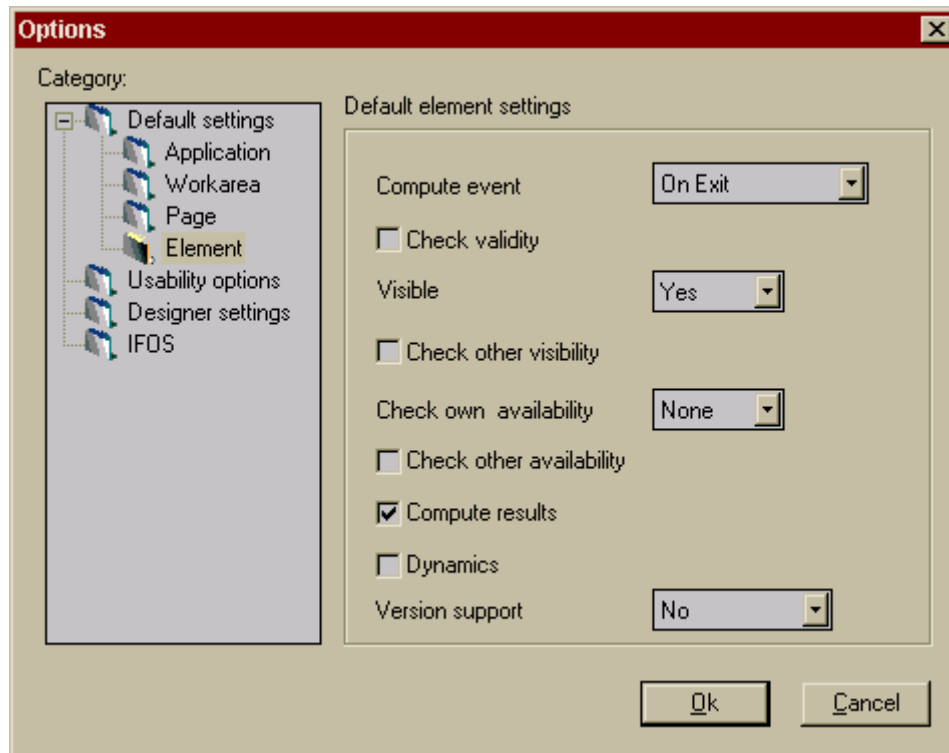
Sets the page sequence to individual or standard.

If page sequence = standard: The element order on a page cannot be changed (the Elements reordering option in the Edit menu is disabled).

Dialog: Options / Page / Version Support

Specifies the default value for the version info property for pages.
Versioning in Designer.
/ v07 X / 000217

Dialog: Options / Element



Dialog: Options / Element / Compute Event

Sets the compute (event) property default for elements.

- On change: Compute default value = OnChange.
- On exit: Compute default value = OnExit.

Dialog: Options / Element / Check Validity

Sets the ChkValidation property default for elements.

Dialog: Options / Element / Visible

Sets the Visible property default for elements.

Note: Whereas the Visible property can be set to Check, the default cannot.

Dialog: Options / Element / Check other visibility

Sets the ChkOtherVisibility property default for elements.

Dialog: Options / Element / Check Own Availability

Sets the ChkOwnAvailability property default for elements.

Dialog: Options / Element / Check other availability

Sets the ChkOtherAvailability property default for elements.

Dialog: Options / Element / Compute results

Sets the ComputeResult property default for elements.

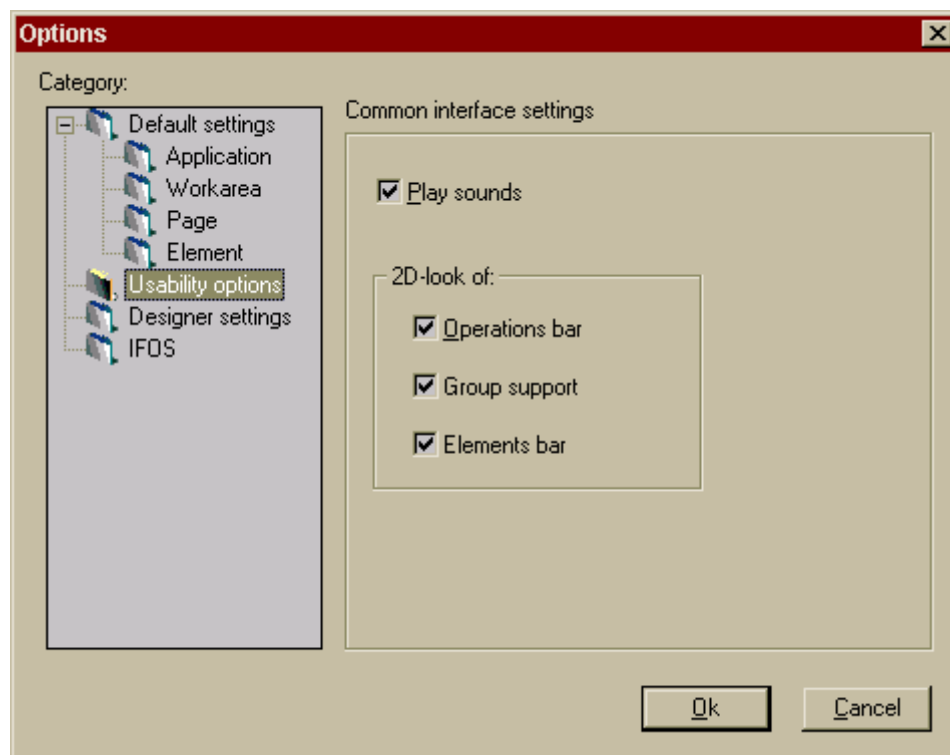
Dialog: Options / Element / Dynamics

Sets the ChkDynamic property default for elements.

Dialog: Options / Element / Version Support

Specifies the default value for the version info property for elements.

Dialog: Options / Useability



Dialog: Options / Useability / Play Sounds

Determines whether or not audio for multimedia elements will be played.

Dialog: Options / Useability / 2d appearance

Determines whether the toolbars will be displayed in 2d or 3d.

Operations ("palette", "common") bar

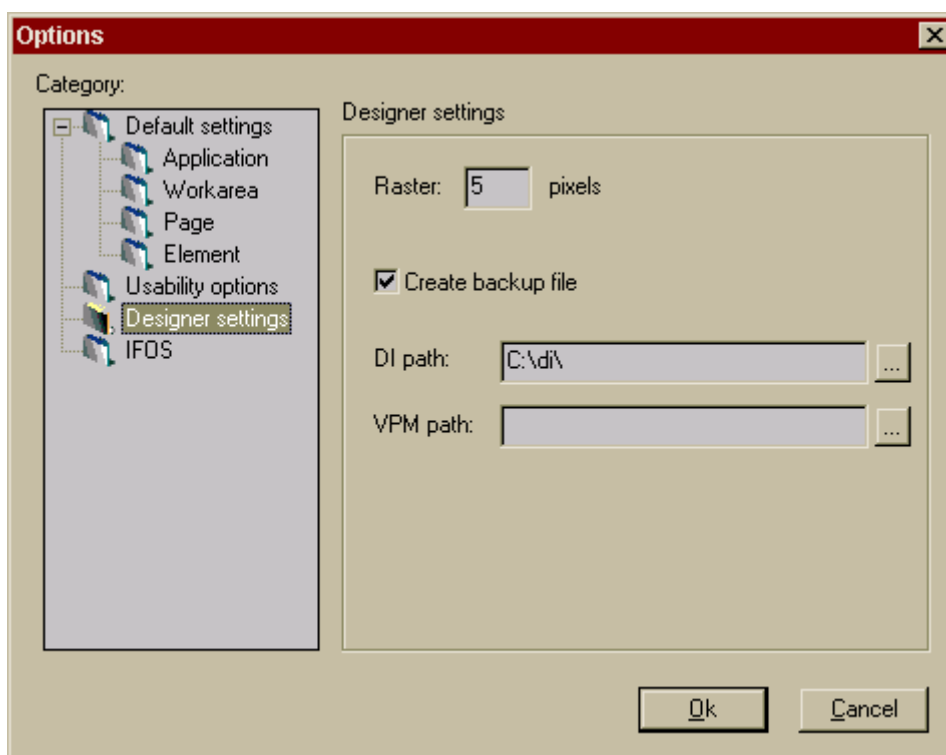
2d



3d



Dialog: Options / Designer settings



Dialog: Options / Designer settings / Raster

Changes the raster settings.

Dialog: Options / Designer settings / Create backup file

Creates a backup file of the layout.

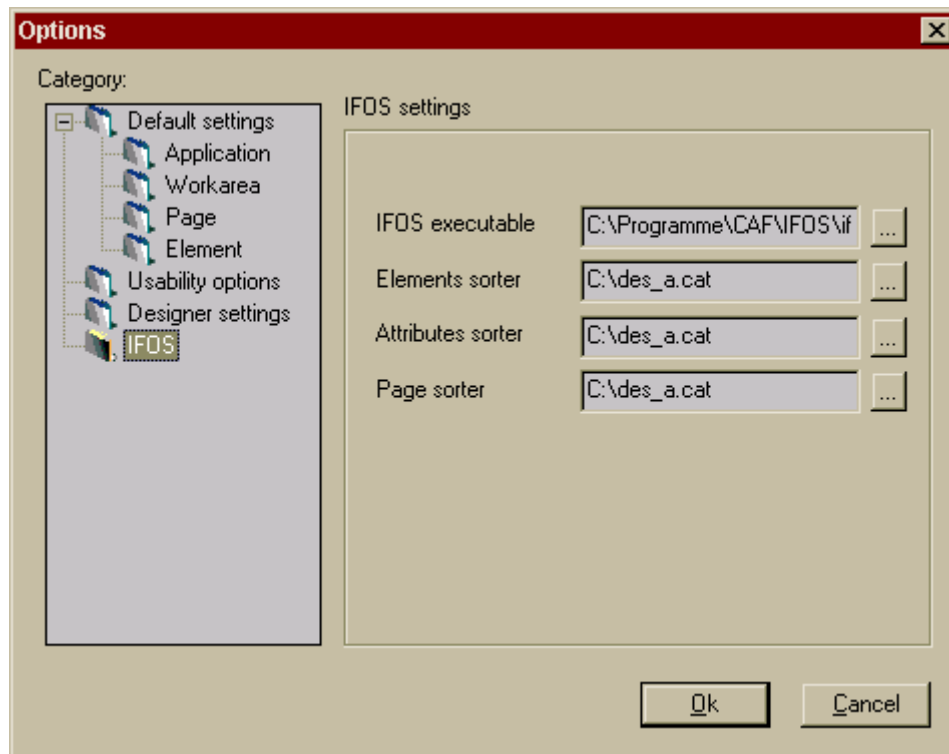
Dialog: Options / Designer settings / DI Path

Specifies the default directory for DI (data indicator) bitmaps.

Dialog: Options / Designer settings / VPM Path

Specifies the default path of .vpm (compiled model) files.

Dialog: Options / IFOS



Dialog: Options / IFOS / IFOS Executable

Specifies the directory / filename of the IFOS executable (normally C:\Programme\CAF\IFOS\ifosre.exe).

Dialog: Options / IFOS / Elements Sorter

Specifies the directory / filename of the elements sorter file.

Dialog: Options / IFOS / Attributes Sorter

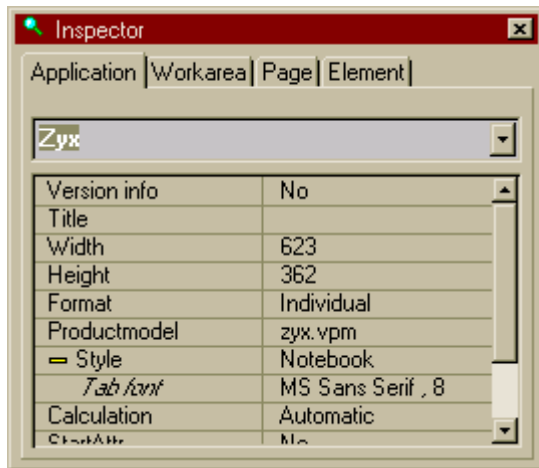
Specifies the directory / filename of the attributes sorter file.

Dialog: Options / IFOS / Page Sorter

Specifies the directory / filename of the page sorter file.

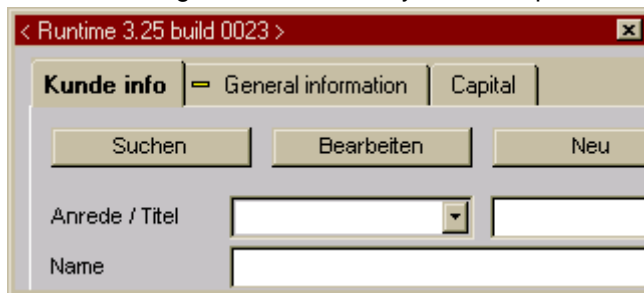
Inspector

The Inspector is used to display the available properties for the currently selected application, workarea, page, element.



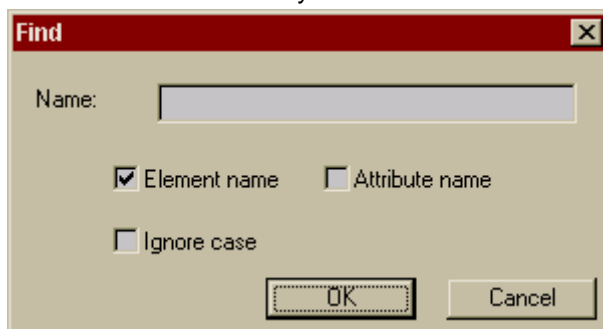
Test dialog

The Test dialog shows how the layout would perform in a consultation.



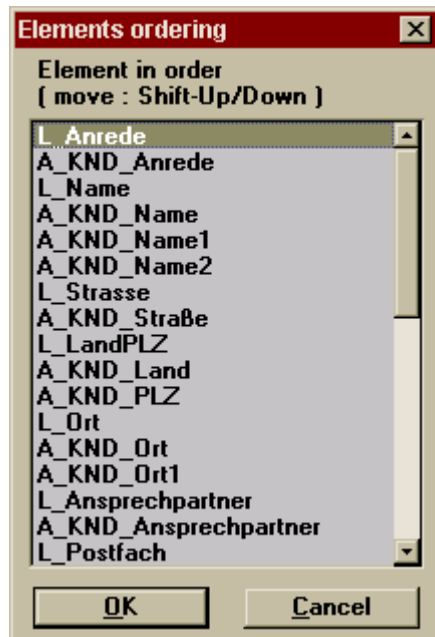
Dialog: Find / Find next

Locates text within the layout.



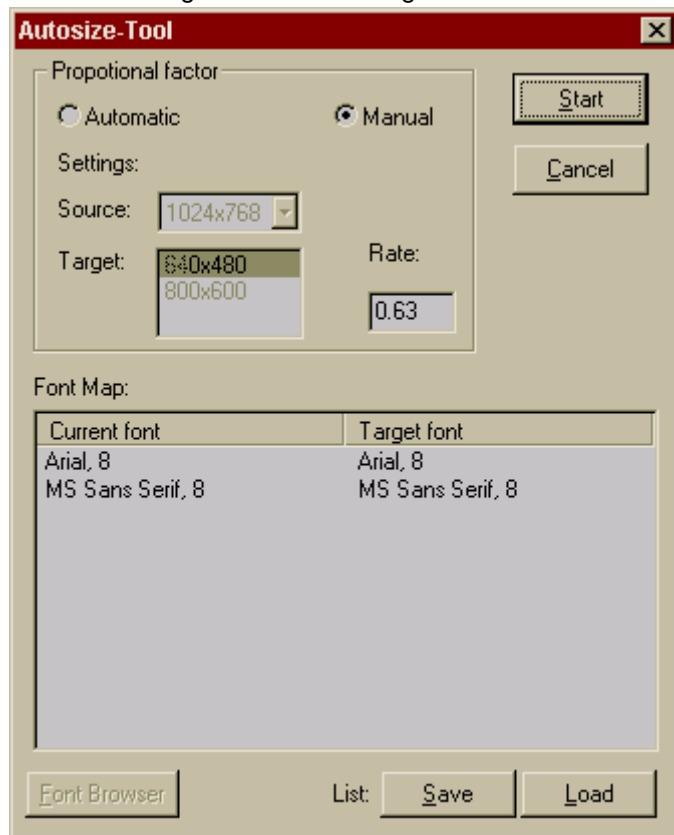
Dialog: Elements (re)ordering

This dialog is used to reorder the elements on a page.



Dialog: Autosize-Tool

This dialog is used to generating a new layout from the current application layout for the target resolution using defined autosizing data.



Dialog: Autosize / Proportional Factor Automatic

Allows the layout to be autosized by simply specifying the target screen configuration.

Dialog: Autosize / Proportional Factor Manual

Allows the layout to be autosized by specifying the target screen size relative to the current screen size.

Dialog: Autosize / font Map

The font maps displays how the fonts will be replaced in the generated layout.

Autosize fontBrowser

The font browser is a standard browser for fonts.

Autosize fontList Save/Load

Click Save to save the font list to a file.

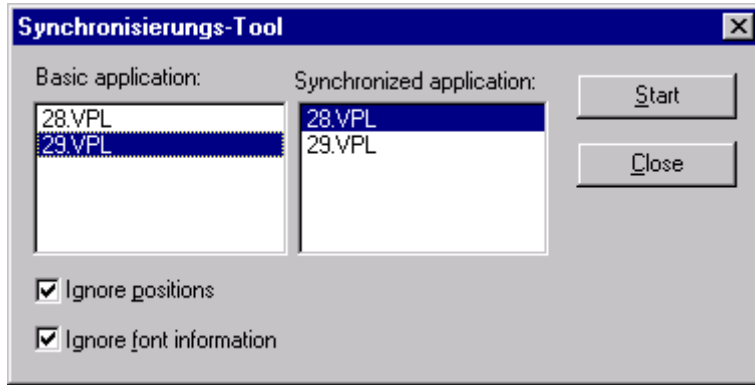
Click Load to load the font list from a file.

Dialog: Synchronisierungs-Tool

The dialog is aimed on getting a report as a result of comparison of two applications opened in different MDI windows (mainly used for synchronizing XPL-applications with different screen resolutions). This dialog can be activated if at least two applications are opened in the Designer.

The dialog includes the following controls.

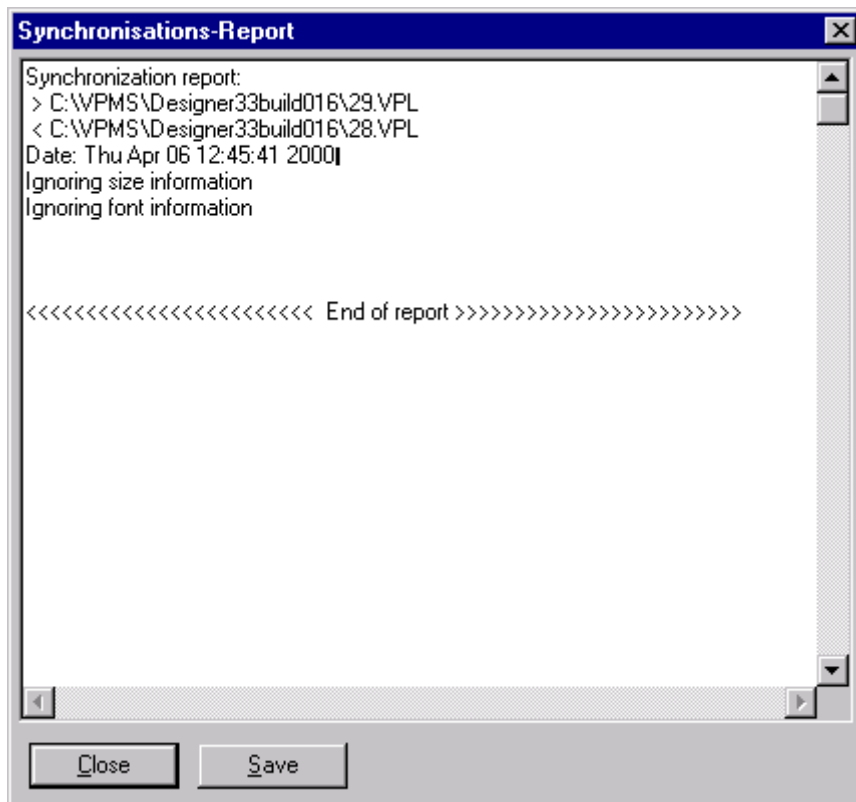
- List box Basis Application. Used for defining a basic application for synchronization. Consists of the list of open applications. Default value is the current (active) open application.
- List box Synchronized Application. Used for defining an application to be compared with the basic application. Consists of the list of open applications excluding the basic application. Default value is the first application of the list.
- Pushbutton Synchronize. Used to generate a synchronization report of the applications defined by the listboxes described above. The generated report is opened in the new MDI window.
- Checkbox Ignore position/size information. Used to avoid reporting differences on elements positioning and size. Default value is checked.
- Pushbutton Close. Used for closing Synchronize dialog.
- Pushbutton Help. Used for getting help information on Synchronize dialog.



Dialog: Synchronisations-Report

Synchronization report windows appears when pushbutton 'Synchronize' in 'Synchronize' dialog is pressed. It is aimed on displaying of synchronization results and provides possibility to save report.

Window overview



Synchronization windows contains a Report display field. In this multi-line scrollable (vertical and horizontal) display field results of synchronization (comparison) are displayed. The report displaying format consists of the header and difference parts.

- The header part contains general information: the names of synchronized applications, synchronization date and time.
- The difference part contains the differences between the applications layouts. For every difference a difference object (in hierarchy area:page:element:property) and the difference character (object does not exist or different properties values) are reported. Values found in the basic application are prefixed with '>' character, values from the synchronized application

are prefixed with '<' character. If an object does not exist in one of the applications, 'not exist' string is reported prefixed with correspondent character.

The Synchronization window also contains:

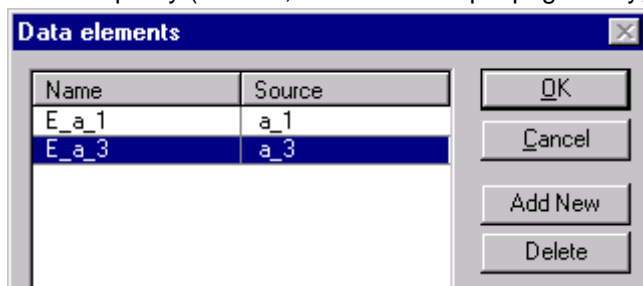
- Pushbutton 'Save'. Used for synchronization report saving. It activates standard 'Save File' dialog.
- Pushbutton 'Close'. For closing synchronization report window.
- Pushbutton 'Help'. Used for getting help information on synchronization report window.

Dialog: Data elements

Data elements support is aimed to provide possibility to have in XPL-application non-visible data elements that are connected to product model attributes. These elements as well as visual elements participate in the data exchange via XPL-data block and their values can be used for computing and setting visual elements values using certain conversion rules (properties) defined in the product model. This conversion process may be activated while running XPL application by some events, such as pushbutton or radio-button pressing or check-box checking.

Window overview

Using 'Data Elements' secondary dialog for every data element the following properties are defined: name (unique in XPL-application among all elements, including visual ones), source (attribute name) and multiplicity (Yes/No, active for multiple pages only, for static pages always No).

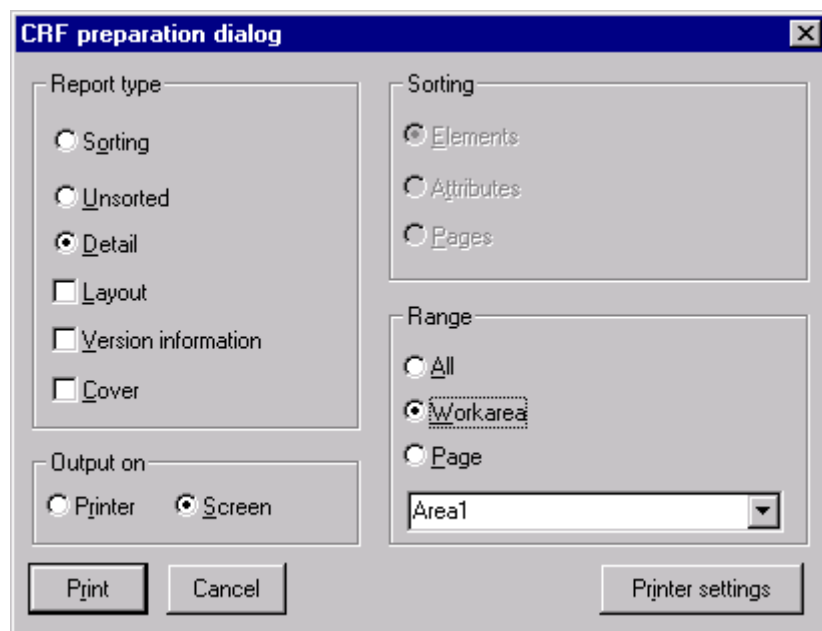


'Data Elements' dialog consists of the following controls.

- Grid 'Data Elements'. The grid contains a data element list defined for the page. For static pages the grid has two columns corresponding to the data elements properties: Name, Source. For multiple pages the grid has three columns: Name, Source, Multiple. Every line of the grid describes a data element. Data elements are ordered in the grid in order of their appearance in the grid. Grid is vertical scrollable. Editing of the data element properties may be made by double-click on the grid cells. Double-click on Name cell causes appearance of an edit-field in this cell containing data element name for editing. Double-click on Source cell causes appearance of a list-box in this cell containing product model attributes to select an attribute for the data element. Double-click on Multiple cell (for multiple pages only) causes appearance of a list-box in this cell containing two options: 'Yes' and 'No'. On right click on the grid area a pop-up menu appears, consisting of two options: 'New' and 'Delete'. Choosing of 'New' option causes addition of a new data element to the grid with automatically generated unique name, which is available to edit. 'Delete' option is used for deletion of the focused data element from the grid producing a warning message-box for confirmation of the action.
- Pushbutton 'New'. Function of this pushbutton is the same as option 'New' in the pop-up menu described above.
- Pushbutton 'Delete'. Function of this pushbutton is the same as option 'Delete' in the pop-up menu described above.
- Pushbutton 'Ok'. When this pushbutton is pressed, all the changes been made in the data elements list are committed and the secondary dialog is closed.

- Pushbutton 'Cancel'. When this pushbutton is pressed, all the changes been made in the data elements list are canceled (the data elements list has the state as before opening the dialog) and the secondary dialog is closed.
- Pushbutton 'Help'. Used for getting help information on 'Data Elements' secondary dialog.

CRF preparation (print) dialog



Report type

"Report type" includes options for selecting the type of report and the content of the report. The following options are available:

- Sorting: Create a table report with entries order by page, element, or element property <Source> value.
- Unsorted: Create a table report with entries in the order as they appear in the layout.
- Detail: Create a text report.
- Layout (available only if "Detail" selected): Include layout information in the report.
- Version information (available only if "Detail" selected): Include component comments ("version information) in the report.
- Cover (available only if "Detail" selected): Include cover information in the report.

Sorting

"Sorting" includes options for selecting the order of the entries in a table report. The following options are available:

- Elements: Order the entries by element name.
- Elements: Order the entries element property <Source> value.
- Elements: Order the entries by page name.

Range

"Range" includes options for selecting the content of a text report.
The following options are available:

- All: Properties for all layout components.
- Workarea: Properties for:
 - Application
 - Selected workarea
 - All components in the selected workarea.
- Page: Properties for:
 - Application
 - Workarea of selected page.
 - Selected page.
 - All components in the selected selected page.

Output On

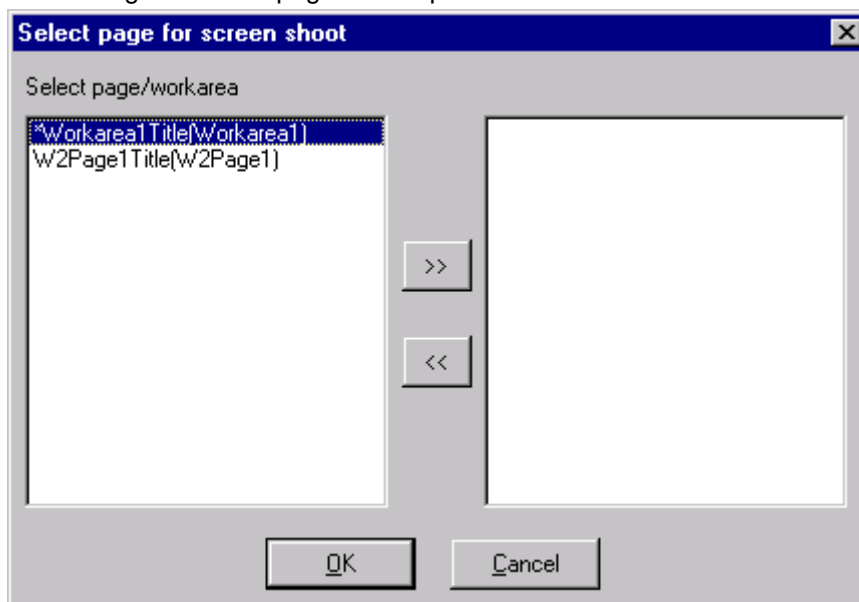
Determines the output device (printer or screen) for the printing.

Printer Settings

Open a dialog for specifying the printer settings for the print.

Select page for screen shot dialog

Used to select the workareas and/or pages to be saved to individual .bmp files.
Opened from the menu Fileselection Export screen shot.
See saving workareas/pages to .bmp files for details.



Concepts

The following are the major concepts for Designer.

- VP/MS components. Describes the VP/MS components (which include Designer).
- Layout components. Describes the components that a layout can contain.
- Add / delete components. Describes how to add and delete layout components.
- Properties. Describes the central function of component properties in the layout.
- Data connections model / layout. Describes the connection between model [attributes] / [attribute properties] / [product properties] and the layout.
- Data types. Describes the data types supported in the layout.
- Entering / selecting data. Describes the various methods Designer provides for entering (EntryField, etc.) and selecting (ComboBox, etc.) data. Also describes other aspects of data entry, including element order, start attribute, the parameter search algorithm and data indicators.
- Displaying data. Describes the methods Designer provides for displaying data.
- Computing result (model property). Describes the different ways of controlling when a result is evaluated in the model and displayed in the layout.
- Re-computing attribute defaults and labels. Describes the different ways of controlling when the attribute properties <default> and <label> are re-computed.
- Visibility. Describes visibility of layout components in the design-time layout (levels) and in the runtime layout (static and dynamic).
- Availability. Describes static and dynamic availability (spelled "availability" in Designer) of layout components. A non-available component is grey and cannot be used for input (although output is supported).
- Subdialogs. Describes how to open and close subdialogs.
- Media element. Describes how to use media elements in a layout.
- DLL calls. Describes how to call DLL's from the layout.
- Help and tooltips. Describes how to implement Windows help and static/dynamic tooltips.
- Layout comment / print. Describes how to comment layout components and create printed lists of layout components.
- Layout test. Describes how to test a layout.

VP/MS components

The following are the major components of VP/MS:

- Model. The model is the source for creating a runtime model.
- Runtime model. The runtime model contains the business logic for the VP/MS application (calculations for the VP/MS application are carried out in the runtime model).
- Layout. The layout is the source for creating a compiled layout.
- Compiled layout. The compiled layout provides the user interface (the layout) for the VP/MS application.
- VFrame consultation. The VFrame consultation integrates the runtime model and compiled layout.
- Report. Describes how a report is used to create a report of the information entered into the layout.

Model

The model is the source that describes the business logic of the VP/MS application.

The model is saved to a .pms file.

The model is created in the Workbench (see the Workbench User's Guide or Online Help for more information about the Workbench).

The model is compiled within the Workbench to create a runtime model.

Runtime model

The runtime model is used in the consultation for the actual calculations required for the business logic.

The runtime model is generated from the model using the Workbench.

The runtime model is saved to a .vpm file.

Layout

The layout is the source that describes the user interface of the VP/MS application.

The layout is saved to a .vpl file.

The layout is created in the Designer.

The layout is compiled within the Workbench to create a compiled layout.

Compiled layout

The compiled layout is used in the consultation to create the user interface.

The compiled layout is generated from the layout using the Designer.

The compiled layout is saved to a .vpc file.

The connection between the layout and model is established by entering the (directory and) filename of the runtime model as the value for the layout property Productmodel.

Attributes of the model are available via the layout drop-down lists for properties if the layout is synchronized with the model. If the required model is the selected model in the Workbench, then synchronization can be accomplished from within the Designer via the synchronization menu item.

Consultation

The runtime layout is accessed by a user by opening a consultation.

A consultation is the framework within which the compiled layout is accessed by a user.

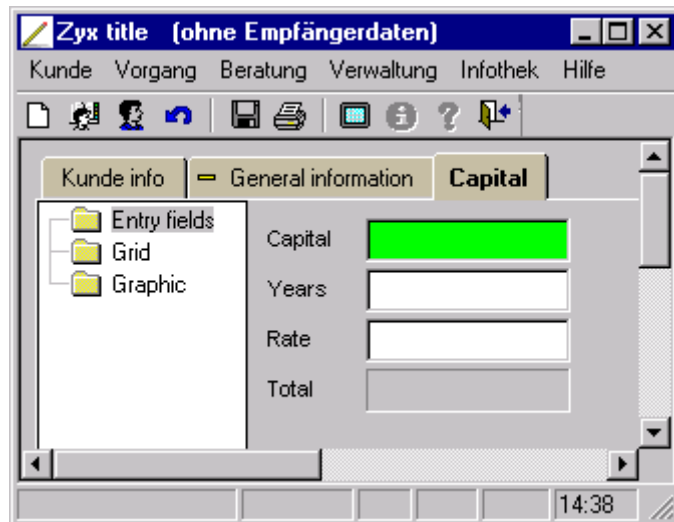
Consult the VP/MS Getting Started manual for more information about the Consultation.

Example

The following shows a VFrame dialog without a consultation ("beratung") selected.



The following shows the VFrame dialog with a consultation (Zyx) selected. The layout displayed is generated by the runtime layout.



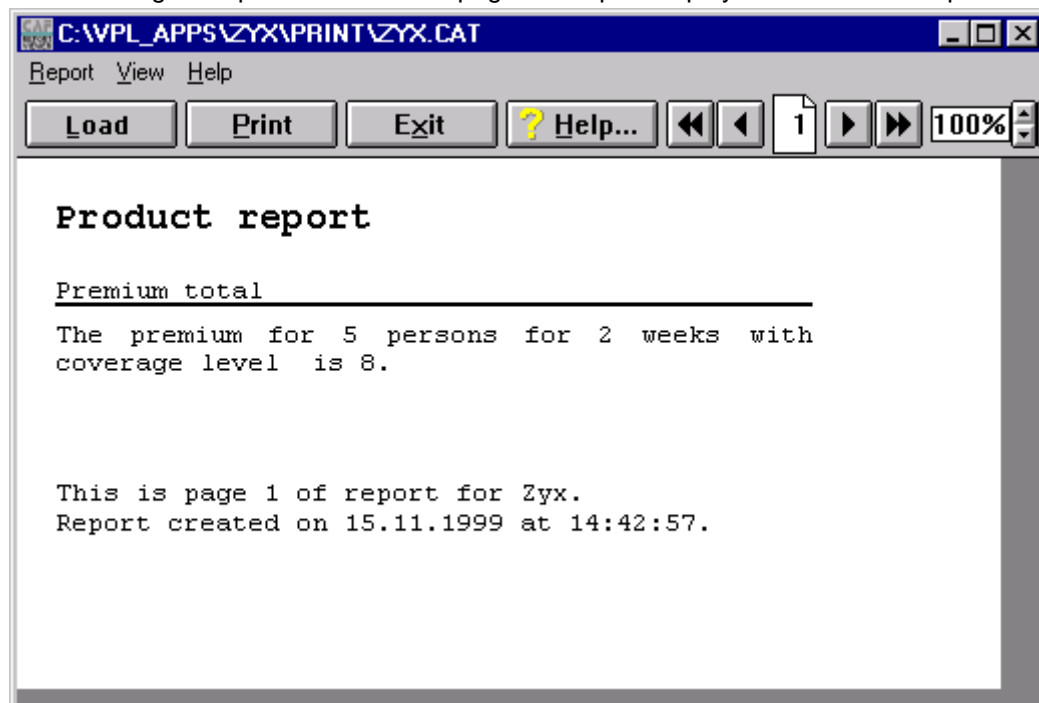
Report

The report is generated with the IFOS Report Generator (for more information about IFOS, consult the IFOS User's Guide).

The report contains the information entered in the consultation. The report can be printed to a file or an output device (printer, screen).

Example

The following example shows the first page of a report displayed in the IFOS Report Viewer.



Layout components

A layout is comprised of an application.

An application can have a header and/or footer.

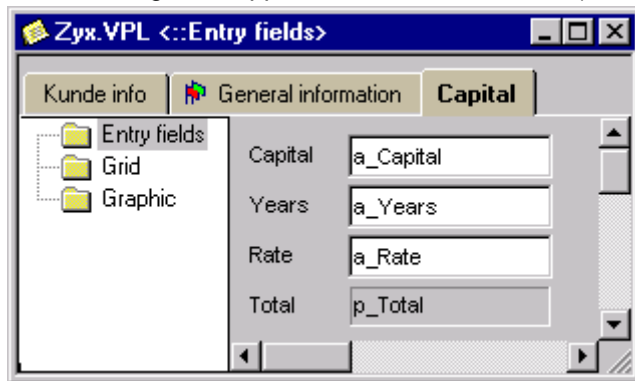
The application contains 1 or more workareas (each with a separate tab).
 A workarea can have a header and/or footer.
 Each workarea contains 1 or more pages.
 Each page contains 0, 1 or multiple elements.

Application

The application is a complete layout.

Example

The following is an application with 3 workareas ("Kunde info", "General information", "Capital").



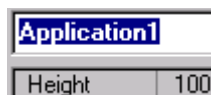
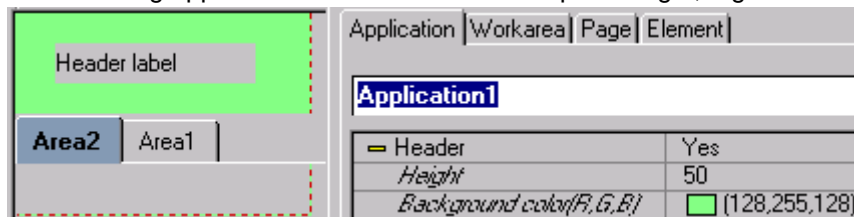
Application Header/Footer

A header / footer can be added to an application with `<Style> = <Notebook>` by defining application properties `<Header> / <Footer>` to `<Yes>`.

Note: An empty (containing no elements) header or footer will not be saved when saving the model.

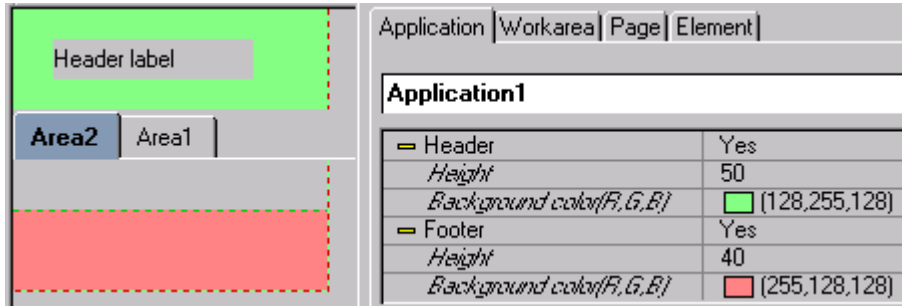
Examples

The following application has a header with 50-pixel height, a green background and a Label.



Note that the application `<Height>` includes the `<Height>` of the header.

The following layout contains a footer. Note that the application `<Height>` also contains the height of the footer.



Application1	
Header	Yes
Height	50
Background color(R, G, B)	(128,255,128)
Footer	Yes
Height	40
Background color(R, G, B)	(255,128,128)

Application1	
Height	140

An application has 1 or more workareas.

If an application has more than 1 workarea, then each workarea has a tab.

If a workarea has more than 1 page, then the left area of the workarea contains a frame with an icon for each page (similar to the tabs for each workarea).

Workarea Header/Footer

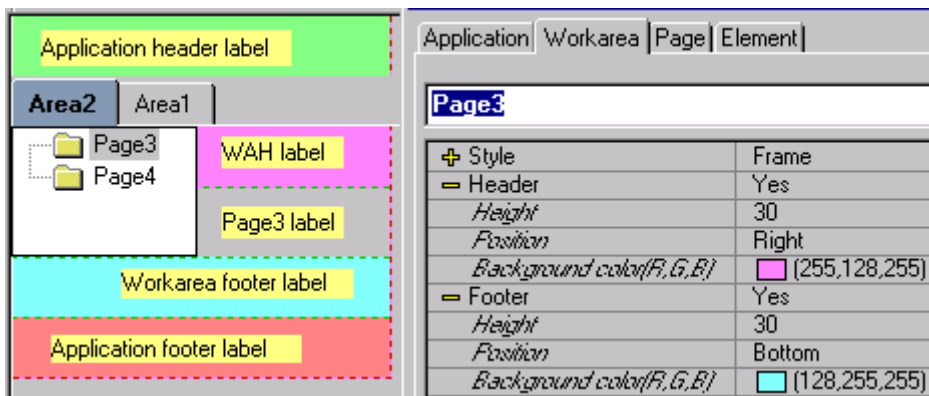
A header / footer can be added to a workarea with <Style> = <Frame> by defining workarea properties <Header> / <Footer> to <Yes>.

Note: An empty (containing no elements) header or footer will not be saved when saving the model.

Example

The following application has:

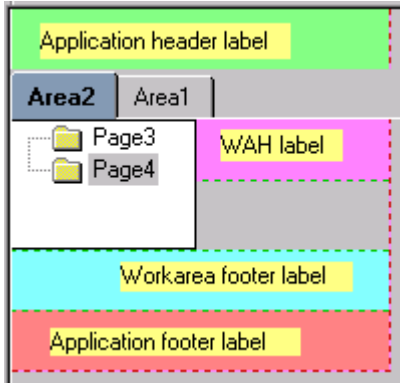
- Workarea Area2 header (pink) with label "WAH label" positioned to the right of the frame.
- Workarea Area2 footer (light blue) with label "Workarea footer label" positioned on the bottom (under) the frame.



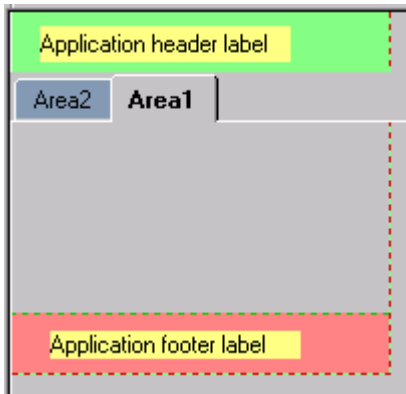
Page3	
Style	Frame
Header	Yes
Height	30
Position	Right
Background color(R, G, B)	(255,128,255)
Footer	Yes
Height	30
Position	Bottom
Background color(R, G, B)	(128,255,255)

The application height includes all footers and headers.

Note that the workarea header/footer is also for Page4.



The workarea Area2 header/footer are not included in workarea Area1.



The following layout has the workarea header <Position> = <Top> and workarea footer <Position> = <Right>.



Page3	
Tooltip	No
+ Style	Frame
- Header	Yes
Height	30
Position	Top
Background color(R, G, B)	□ (255,128,255)
- Footer	Yes
Height	30
Position	Right
Background color(R, G, B)	□ (128,255,255)

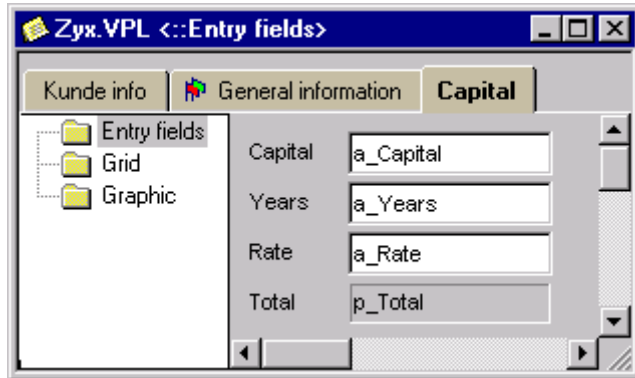
Page

Each workarea has 1 or more pages.

Only 1 page can be displayed in the layout at any time (the frame with the name of other pages for the selected workarea and the tabs with the names of all workareas for the application are also displayed). A page contains 0, 1 or multiple elements.

Example

In the following example, the workarea with tab text "Capital" has a frame with the names of 3 pages ("Entry fields", "Grid", "Graphic"). Page "Entry fields" is the currently displayed page.



Element

A page can contain any number of elements.

An element can be used to:

- Enter/select data
- Display data
- Make external calls

The following types of elements are available in Designer:

- EditField
- ComboBox
- CheckBox
- RadioButton
- PushButton
- Label
- Border
- Grid
- ExtendedGrid
- 3D Border
- Line
- Multimedia element
- ResultField
- RadioGroup
- Graphic

Example

In the following example, the workarea with tab text "General information" has many elements (labels, entry fields, combo box, multimedia, border, 3d border, etc.), each with a different color.

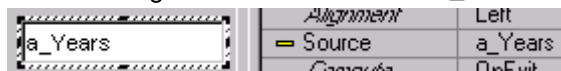


Concept: EditField

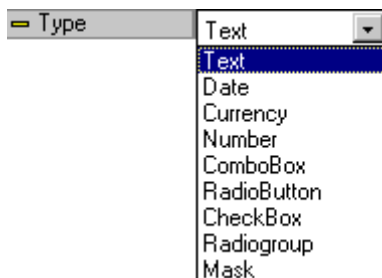
An EditField is used to enter data. An EditField can also be used to dynamically display data. The Sourceproperty of an EditField specifies the model attribute or model property (not to be confused with a designer component property) that is entered / displayed in the EditField. The Typeproperty specifies the data type of the attribute.

Example

The following EditField has attribute a_Years as the source.



The type of the entered data is by default Text. Note that not only the type of data can be specified (Text / Date / Currency / Number), but also that the type of element can be changed (to ComboBox / RadioButton / CheckBox / RadioGroup / Mask).



Concept: ComboBox

A ComboBox is used to enter data(as for an EditField) or select data (from a list of available values). The Source property of a ComboBox specifies the model attribute or model property (not to be confused with a designer component property) that is entered / selected / displayed in the ComboBox.

Example

The following ComboBox has attribute a_Weeks as the source.



During runtime, the ComboBox can be used to display the available values (defined in a Workbench table) as shown below (the available values are "1 week" and "2 weeks"):



Concept: CheckBox

A CheckBox is used to select a value of type boolean.

The Source property of a CheckBox specifies the model attribute or model property (not to be confused with a designer component property) that is selected / displayed in the CheckBox.

Example

The following CheckBox has attribute Zuweisen as the source.



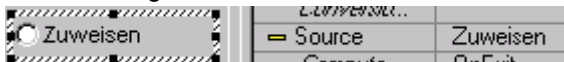
Concept: Radio button

A RadioButton is used to select a value of type boolean.

The Source property of a RadioButton specifies the model attribute or model property (not to be confused with a designer component property) that is selected / displayed in the RadioButton.

Example

The following RadioButton has attribute Zuweisen as the source.



Concept: Push Button

A PushButton is used for an external calls.

The Action property of a PushButton specifies the action (Compute, DLL, Dialog, Accept, Cancel) taken when the PushButton is clicked.

Example

The following PushButton makes a call to the DLL specified in the the file kndbearb.ini when clicked.



Concept: Label

A Label is used to display static text or dynamic text.

The text displayed by a Label is specified by:

- IF property Attribute is not defined: Property Title.
- IF property Attribute is defined: The text string referenced the attribute.

Example

The text displayed by the following Label is defined by the model attribute A_KND_Geschlecht.

Ansprechpartner	Type	Label
	Title	any_text
	Default font	
	Attribute	A_KND_Geschlecht

Concept: Border

A Border is used to visually (not functionally) group together elements.

A Border is also used to display (in the upper-left corner of the Border) static text or dynamic text.

The text displayed by a Border is specified by:

- IF property Attribute is not defined: Property Title.
- IF property Attribute is defined: The text string referenced the attribute.

Example

The text displayed by the following Border is defined by the Border title property.

Required information	Type	Border
	Title	Required information
	Default font	

Concept: Grid

A Grid is used to display the model attributes and model properties for an array of elements.

A Grid consists of the following:

- Columns
- Rows

Grid columns support certain data types.

Example

The following example shows a layout Grid with 4 columns (column 2 is used to generate the array pointer).

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4

Type	Grid
Column	4
Columns moveable	Yes
Key column	1
Iteration type	Attribute
Attribute	a_Years

The following shows the above Grid in a compiled layout with an array of years with 3 members:

Year	(internal)	Interest	Capital
1	2	20 DM	120 DM
2	3	24 DM	144 DM
3	4	28 DM	172 DM

Concept: Grid Columns

A Grid normally has 3 types of columns:

- A single key column that displays the array key
- A hidden (property Width = 0) next-key column used for generating for the key for the next row
- 1 or more attribute or property columns

Concept: Grid Key Column

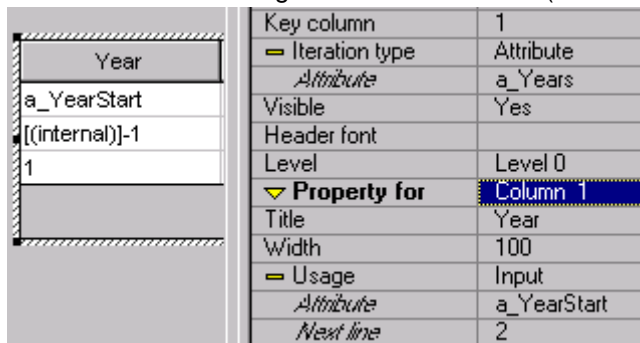
The Key column (usually the first column) uniquely identifies each row.

The following are the most important properties for the key column:

- "Usage". Normally set to "Input", because the key column row will receive input (the text to display in a cell).
- "Attribute". Attribute is the model attribute that determines the value for the key column in the first row.
- "Next line". Specifies the column in the grid which defines how the text for the key column in the next row is generated.

Example

In the following Grid, Attribute "a_YearStart" provides the value for the key column in the first row. The text for the next row is generated in column 2 ("Next line").



The result is shown in the following runtime layout:

Year	(internal)
1	2
2	3
3	4

Concept: Grid Next Key Column

The Next Key column (typically the last column; however, it can be any column) provides the link to the runtime model property that generates the text for the key column for the next row.

The following are the most important properties for the next key column:

- "Usage". Normally set to "Output", because the next key column row will output data for the next row's key column.
- "Property". Property is the model property that generates the value for the key column in the next row.

Example

In the following Grid, column 2 (the next key column) is linked the model attribute p_YearNext(). Column 1 specifies Column2 as the source for the Column 1 text ("Next line"). Note in the column Year in the second row the text "[((internal))-1]": This specifies that the key column text is generated in the column with title "(internal)" in the preceding ("-1") row.

Year	(internal)
a_YearStart	p_YearNext()
[((internal))-1]	p_YearNext()
1	2

▼ Property for Column 2	
Title	(internal)
Width	100
Usage	Output
Property	p_YearNext()

Concept: Grid Attribute / Property Columns

The remaining (ie, not Key Column or Next Key Column) columns display the value in an array. The following are the most important properties for attribute / property columns:

- "Usage". Normally set to "Output".
- "Property". Specifies the model property that references an array of data.

Note also the property under "Iteration type". This specifies the index attribute in the model for the arrays of data.

Example

In the following Grid, column 3 contains data values from the array "p_Interest()" in the model. The index for this array in the model is "a_Years". The value displayed in a row in the Grid for Column "Interest" is determined by the value in the Key Column of the Grid. This value is used as the index a_Years for looking up the value.

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[((internal))-1]	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4

Iteration type	Attribute
Attribute	a_Years
Visible	Yes
Header font	
Level	Level 0
▼ Property for Column 3	
Title	Interest
Width	100
Usage	Output
Property	p_Interest()
Type	Currency

In the following runtime layout example:

- p_Interest(1) = DM 22
- p_Capital(1) = DM 122
- p_Interest(2) = DM 26
- p_Capital(2) = DM 148

Year	(internal)	Interest	Capital
1	2	22 DM	122 DM
2	3	26 DM	148 DM
3	4	32 DM	181 DM

Concept: Grid column data types

The data displayed in a Grid column can be of the following types:

- Text

- Number
- Currency
- Date
- MaskResult

Concept: Grid Rows

Each Grid row shows the members of all of the arrays in the Grid with an index equal to the value in the Key column for that row.

Concept: ExtendedGrid

An ExtendedGrid is visually similar to a Grid. However, the functionality of the 2 element types differs significantly.

The major difference is that data can be entered (not simply displayed) in an ExtendedGrid.

The property Count attribute specifies the model attribute that will be used to count the rows.

An ExtendedGrid has 2 types of columns:

- Input. An Input column can only reference a model attribute.
- Output. An Output column can only reference a model property.

The ExtendedGrid rows are similar in function to those of the Grid with one major exception: Rows can be dynamically added or deleted in an Extended Grid.

For step-by-step examples of using the ExtendedGrid, consult the VP/MS Getting Started Guide.

Example

(The following example is used in the help subtopics for this topic).

The following example shows a layout ExtendedGrid with 4 columns.

- Columns 1 and 4 are output columns (referencing model properties).
- Columns 2 and 3 are input columns (referencing model attributes (column 2 uses a ComboBox for the input)).

Nr	Name	Date of birth	Premium
Folge()		a_DOB	p_PremiumP2Each()
1		3	4

The following shows the above ExtendedGrid in a compiled layout:

Nr	Name	Date of birth	Premium
1	name1	11.11.1811	1.000 DM
2			0 DM

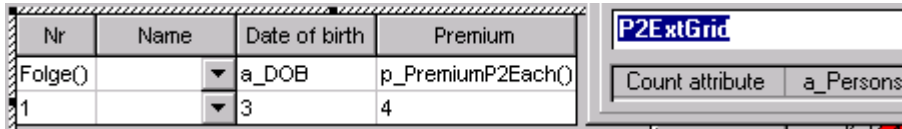
ExtendedGrid: Count attribute

The property "Count attribute" specifies the model attribute that is the count attribute for the multiple inclusion rule in the model (in model terminology, the "IncType" = "multiple" and the "IncRule" = the model attribute).

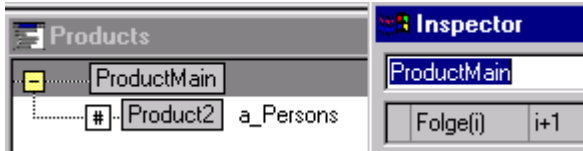
Note: To display the count attribute in the ExtendedGrid, the attribute must be sent from the model to the layout via a property (shown in the example below).

Example

In the following ExtendedGrid, the Count attribute is set to a_Persons. Thus, whenever a new row is added to the ExtendedGrid, a_Persons will be incremented.



In the model, a_Persons is the IncRule for a multiple inclusion. Note that the property Folge(i) is used to send the count attribute back to the layout (the count is incremented by 1, since the index for the array starts at 0).



Note: Any name can be used to reference the index ("i" is used above). Workbench automatically substitutes the IncRule variable into the index variable.

The following runtime layout shows the results:

Nr	Name	Date of birth	Premium
1			0 DM
2			0 DM

Note: No input was entered for the above example. 2 persons are shown because a_Persons has a default of 2.

ExtendedGrid: Column types

An ExtendedGrid has 2 types of columns:

- Input
- Output

ExtendedGrid: Input column

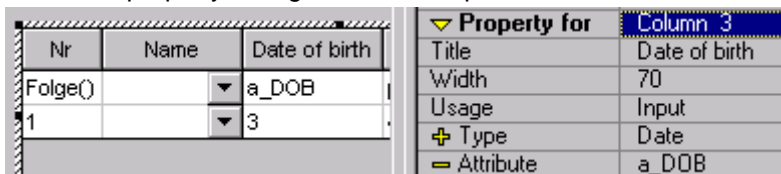
ExtendedGrid input columns support only certain data types.

Note: An ExtendedGrid can accept text input by specifying the property "Type" as a ComboBox.

An input column must reference a model attribute.

Example

Column 3 property "Usage" is set to "Input". The attribute is a_DOB.



Column 2 property "Usage" is set to "Input". The attribute is a_Name. Note that the Type is ComboBox.

Nr	Name	Date of birth
Folge()		a_DOB
1		3

Property for		Column 2
Title		Name
Width		70
Usage		Input
Type		ComboBox
+ Attribute		a Name

In the runtime layout, the values for the indexed attributes are entered directly in the ExtendedGrid.

Nr	Name	Date of birth	Premium
1	name2	11.11.1811	1.000 DM
2			0 DM

ExtendedGrid: Output column

ExtendedGrid output columns support only certain data types.

There are 2 types of output columns in an ExtendedGrid:

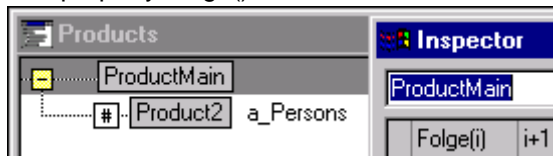
- The counter output column. References a model property in the upper branch of the multiple inclusion.
- A row result column(s). References a model property in the lower branch of the multiple inclusion.

Example

Column 1 property "Usage" is set to "Output". The model property is Folge().

Nr	Property for		Column 1
Folge()	Title		Nr
1	Width		40
	Usage		Output
	+ Type		Text
	+ Property		Folge()

The property Folge() is defined in the model in the upper branch of the multiple inclusion.



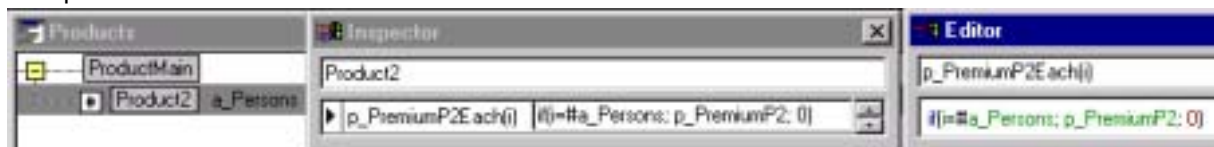
Note that the Workbench automatically inserts the indexed value of the IncRule a_Persons for the index variable ("i").

Column 4 property "Usage" is set to "Output". The model property is p_PremiumP2Each().

Nr	Name	Date of birth	Premium
Folge()		a_DOB	p_PremiumP2Each()
1		3	4

Property for		Column 4
Title		Premium
Width		100
Usage		Output
+ Type		Currency
+ Property		p_PremiumP2Each()

Note, however, that property p_PremiumP2Each() is defined in the model in the lower branch of the multiple inclusion.



The runtime layout results:

Nr	Name	Date of birth	Premium
1			0 DM
2			0 DM

Concept: ExtendedGrid column data types

The data displayed in a Grid column can be of the following types:

- Text (this is also the data type when a ComboBox is selected)
- Number
- Currency
- Date
- MaskResult

Concept: ExtendedGrid Rows

Each Grid row shows the members of all of the arrays in the Grid with an index equal to the value the counter column.

Add/delete ExtendedGrid Rows

Rows can be dynamically added or deleted in an ExtendedGrid.

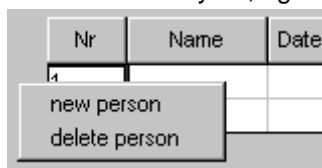
Example

The properties "Caption New" and "Caption Delete" specify the text that will be displayed when right clicking on an ExtendedGrid.

Nr	Name	Date of birth	Premium
Folge()		a_DOB	p_PremiumP2Each()
1		3	4

Count attribute	a_Persons
Caption New	new person
Caption Delete	delete person

In the runtime layout, right clicking will display the following context menu.



Concept: 3d Border

A 3D Border is used to visually (not functionally) group together elements. A 3D Border does NOT display (in the upper-left corner of the Border) text.

Example

The following diagram shows a 3D Border.



Concept: Line

A Line is used to visually (not functionally) separate elements.

Example

The following diagram shows 3 Line elements.



Concept: Multimedia element

A Multimedia element can have as the Source a file of the following types:

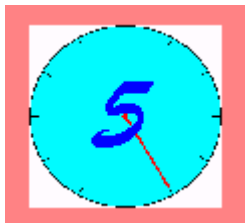
- .avi
- .wav
- .bmp
- .jpg
- .tif

Example

A Multimedia element in the non-runtime layout always appears with the VP/MS icon:



In the runtime layout the appropriate type of multimedia element is displayed/played. In the diagram below, an .avi movie is shown.



Note: The movie plays only when the page of the movie is selected. If another page or workarea is selected, the movie stops playing. If the page is again displayed, the movie starts from the beginning.

Concept: ResultFields

A ResultField is used to display a property that is a result of other properties/attributes.

The type of ResultField must match the data type to be displayed. The following types of ResultFields exist:

- Result (TextResult)
- NumberResult

- CurrencyResult
- DateResult
- MaskResult

Result

The Result field is used to display any type of data as a String.

Example

The following is a Result field.

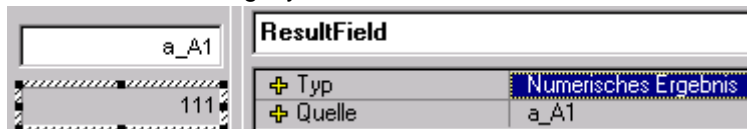


NumberResult

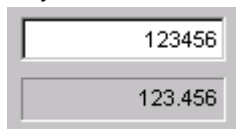
A NumberResult field is used to display numeric results.

Example

Assume the following layout with an EditField and a NumberResult field.



Any data entered in the EntryField will be displayed in numerical format.

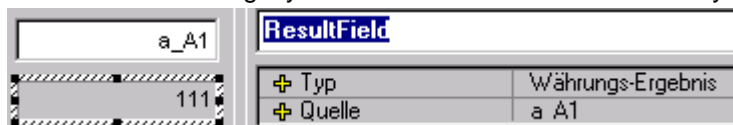


CurrencyResult

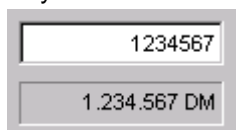
A CurrencyResult field is used to display currency results.

Example

Assume the following layout with an EditField and a CurrencyResult field.



Any data entered in the EntryField will be displayed in currency format.



DateResult

A DateResult field is used to display date results.

Example

Assume the following layout with an EditField and a DateResult field.

<input type="text" value="a_A1"/> <input type="text" value="27.05.97"/>	<table border="1"> <thead> <tr> <th colspan="2">ResultField</th> </tr> </thead> <tbody> <tr> <td>+ Typ</td> <td>Datums-Ergebnis</td> </tr> <tr> <td>+ Quelle</td> <td>a_A1</td> </tr> </tbody> </table>	ResultField		+ Typ	Datums-Ergebnis	+ Quelle	a_A1
ResultField							
+ Typ	Datums-Ergebnis						
+ Quelle	a_A1						

Any data entered in the EntryField will be displayed in date format.

<input type="text" value="1.1.1911"/> <input type="text" value="01.01.11"/>
--

Note that invalid input (input that does not represent a valid date) will not be displayed.

<input type="text" value="12345"/> <input type="text"/>
--

Normally the Date EntryField is used to restrict the values entered for an attribute to valid Date strings.

MaskResult

A MaskResult field is used to display a customized data format using a mask.

Example

Assume the following layout with an EditField and a MaskResult field.

<input type="text" value="a_A1"/> <input type="text" value="a_A1"/>	<table border="1"> <tbody> <tr> <td>- Typ</td> <td>Maske-Ergebnis</td> </tr> <tr> <td><i>Maskierung</i></td> <td>UUNNLL</td> </tr> <tr> <td><i>Ausrichtung</i></td> <td>Rechts</td> </tr> <tr> <td>+ Quelle</td> <td>a_A1</td> </tr> </tbody> </table>	- Typ	Maske-Ergebnis	<i>Maskierung</i>	UUNNLL	<i>Ausrichtung</i>	Rechts	+ Quelle	a_A1
- Typ	Maske-Ergebnis								
<i>Maskierung</i>	UUNNLL								
<i>Ausrichtung</i>	Rechts								
+ Quelle	a_A1								

Any data entered in the EntryField will be displayed in the format defined by the mask.

<input type="text" value="abcdefgh"/> <input type="text" value="AB_cd"/>

<input type="text" value="1234567"/> <input type="text" value="__12__"/>

Normally a Mask EntryField is used to restrict the values entered for an attribute to the required format to be displayed in the MaskResult field.

Concept: RadioGroup

A RadioGroup is used to:

- Display a list of available values
- Select a value from the list of available values

The major property for a RadioGroup is the Source. The Source references a model attribute which references a table with the text for the RadioGroup selections (see the Workbench User's Guide for more information).

Example

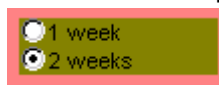
The following RadioGroup in the layout has as Source attribute a `_Weeks`.



Attribute `a_Weeks` references a table (in the model) with the following content:

a_Weeks	
key	value
1	1 week
2	2 weeks

The content is displayed in the runtime version of the RadioGroup:



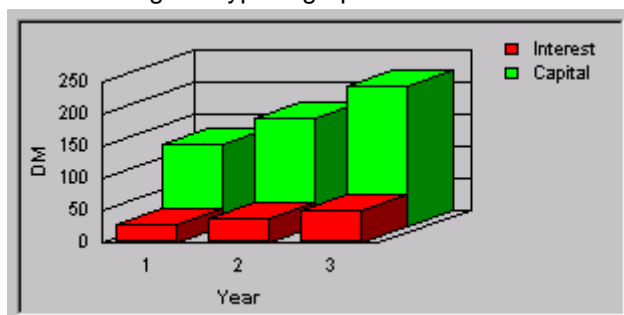
Concept: Graphics element

A Graphics elements can display arrays of data in a graphic format.

- A graphics element requires an ID Grid as the source of information to be displayed.
- The key column provides the numerical values for the x-axis of the graphic.
- A serie is a set of data in the graphic. A graphic can contain multiple series, with each serie corresponding to a column of data in the id grid.
- The normalized result of the series becomes the numerical value for the y-axis.
- The legend is a description of each serie in the graphic.
- The axis names describe the x-axis and y-axis.
- There are several graphic element types.

Example

The following is a typical graphic.



In the above graphic:

- The key column of the source grid contains values 1, 2, 3.
- The graphic displays 2 series of data (a green and a red serie).
- The normalized result range shown in the y-axis is DM 0 ... 250.
- The legend for the red serie is "Interest" and for the green serie "Capital" (column names from the source grid).
- The axis names are "Year" and "DM".
- There graphic type is 3D vertical bar.

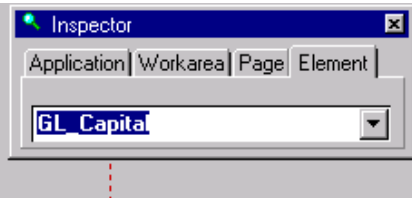
Identification Grid

The identification grid is a Grid or ExtendedGrid that provides the information to be displayed in the graphic. The ID grid is specified by entering the element name of the grid for the graphic property Identification grid.

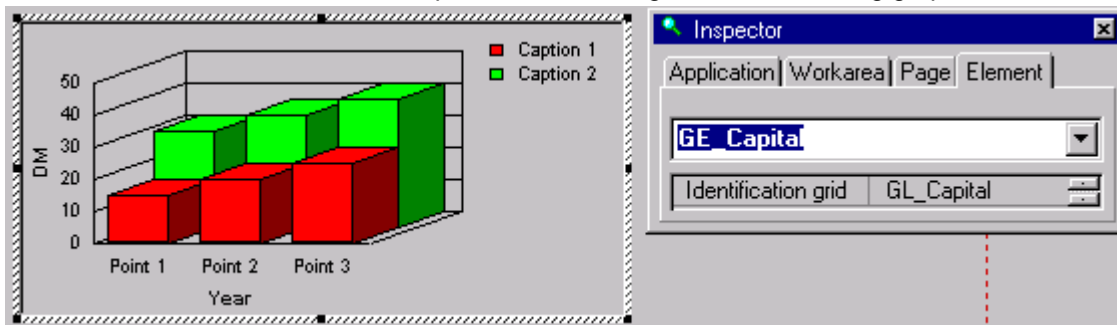
Example

Assume the following ExtendedGrid.

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4



The above ExtendedGrid would be specified as the ID grid for the following graphic.

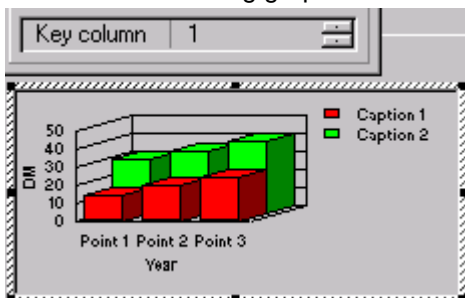


Key column

The key column is a column in the Identification Grid and provides the numerical values for the x-axis of the graphic. The key column is specified with the property Key column.

Example

Assume the following graphic:

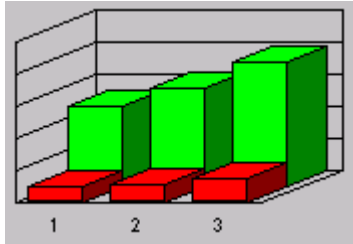


Assume the following Grid with the entered data as the Identification grid:

Year	(internal)	Interest	Capital
1	2	25 DM	125 DM
2	3	31 DM	156 DM
3	4	39 DM	195 DM

Note that the above grid contains entries for 3 years.

The resulting graphic would be:



The digits "1", "2", "3" above is the text from column 1 in the Identification grid.

Serie

A serie is a set of data in the graphic. A graphic can contain multiple series, with each serie corresponding to a column of data in the id grid.

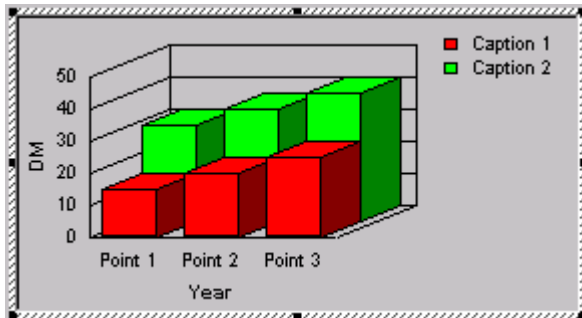
The number of series in a graphic is specified by the property Series.

The data and color of each series is specified by:

- Selecting the serie from the drop-down list of all series in the "Property for" property.
- Specifying the data column in the Identification grid that supplies each value for the serie by specifying property Data column.
- Specifying the color of the serie by specifying property Color.

Example

Assume the following graphic:



Series 2

Assume the following Grid with the entered data as the Identification grid:

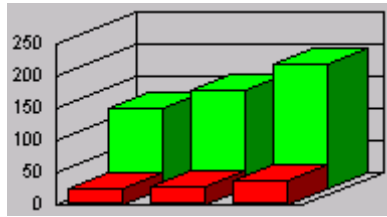
Year	(internal)	Interest	Capital
1	2	25 DM	125 DM
2	3	31 DM	156 DM
3	4	39 DM	195 DM

Note the contents in columns 3 and 4 in the above grid.

▼ Property for	Serie 1
Data column	3
Color	■ (255,0,0)

▼ Property for	Serie 2
Data column	4
Color	■ (0,255,0)

The resulting graphic would be:



Normalized result


The normalized result of the series becomes the numerical value for the y-axis. The range of values is determined automatically by Designer.


Example

Assume the following Grid with the entered data as the Identification grid:

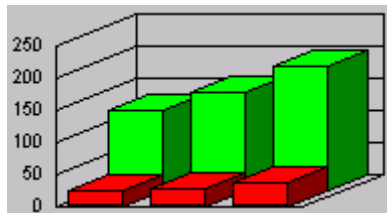
Year	(internal)	Interest	Capital
1	2	25 DM	125 DM
2	3	31 DM	156 DM
3	4	39 DM	195 DM

Note the contents in columns 3 and 4 in the above grid.

▼ Property for	Serie 1
Data column	3
Color	 (255,0,0)

▼ Property for	Serie 2
Data column	4
Color	 (0,255,0)

The resulting graphic would have the range 0..250 on the y-axis.



Legend

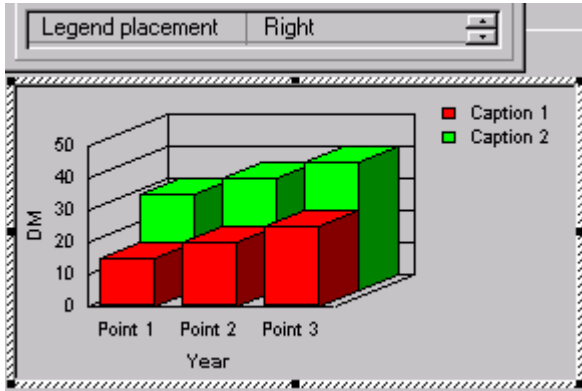
The legend is a description of each serie in the graphic.

The legend is the name of the column in the Grid.

The legend can be placed to the right or bottom of the graphic by specifying property

Example

Assume the following graphic:



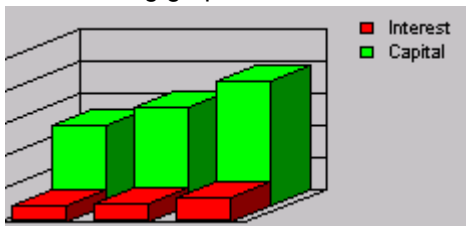
Note "Caption 1" and "Caption 2" above. In the runtime the captions will be replaced by the text heading of the appropriate column in the Grid. Assume the following Grid with the entered data as the Identification grid:

The column name is specified by Grid property Title as shown below for column 3:

▼ Property for		Column 3
Title		Interest

Interest	Capital
p_Interest()	p_Capital()
p_Interest()	p_Capital()
3	4

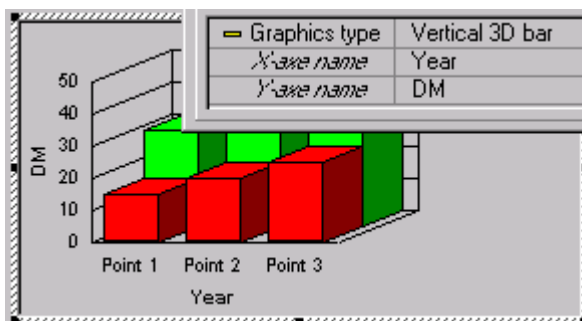
The resulting graphic would be:



Axis names

The axis names describe the x-axis and y-axis and are specified with the properties X-axis name and Y-axis name.

Example



Graphic element types

The following types of graphic elements are available:

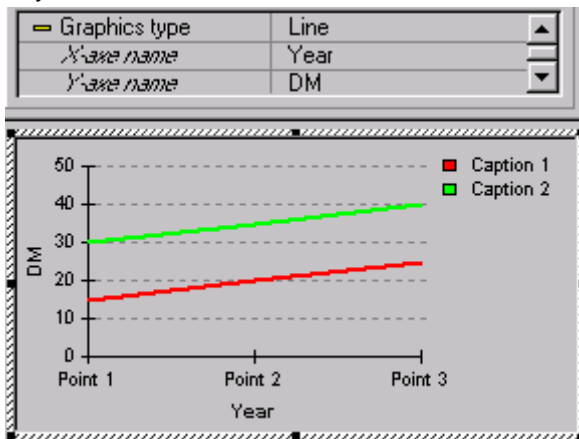
- Line
- Vertical bar
- Horizontal bar
- Vertical 3D bar
- Horizontal 3D bar
- Vertical Gantt
- Horizontal Gantt
- Pie
- 3D Pie

Line

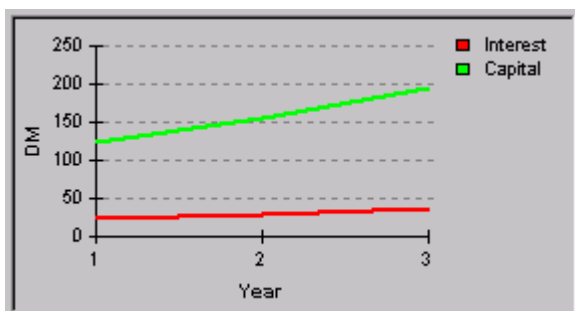
The graphic type line shows each serie as a line.

Example

Layout:



Runtime:

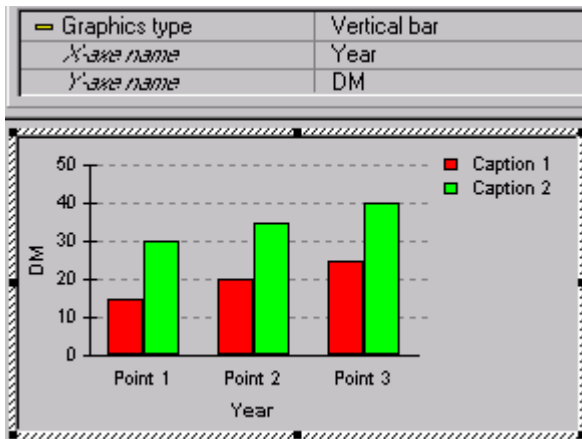


Vertical Bar

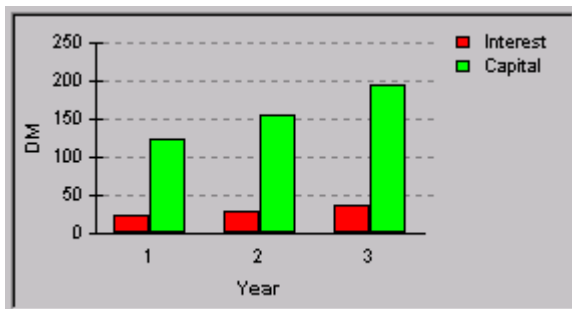
The graphic type vertical bar shows each serie as a vertical bar.

Example

Layout:



Runtime:

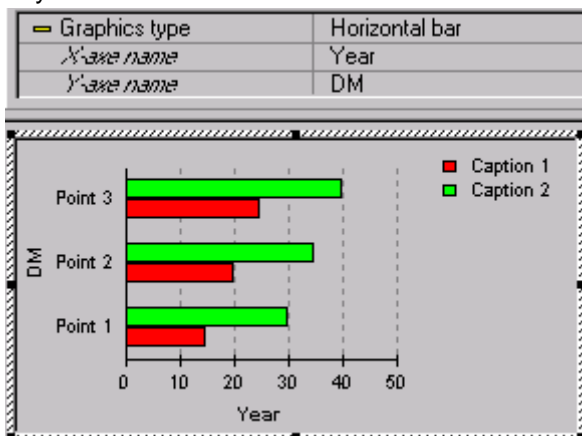


Horizontal Bar

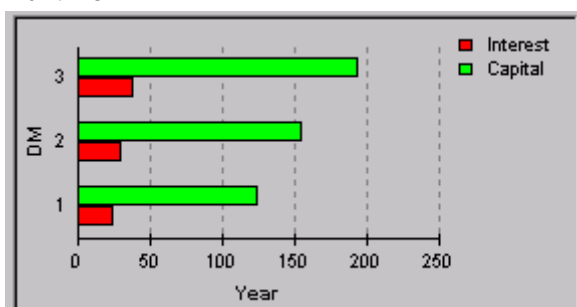
The graphic type horizontal bar shows each serie as a horizontal bar.

Example

Layout:



Runtime:

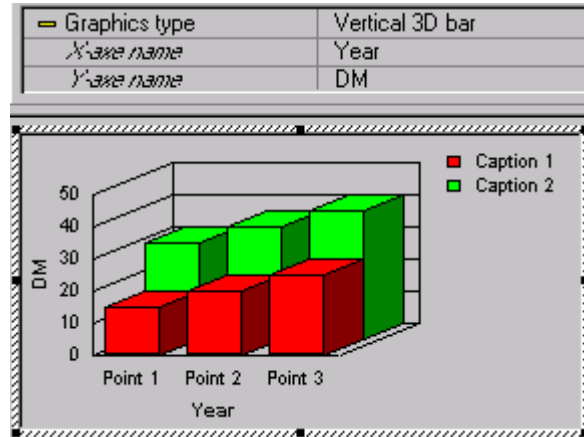


Vertical 3D Bar

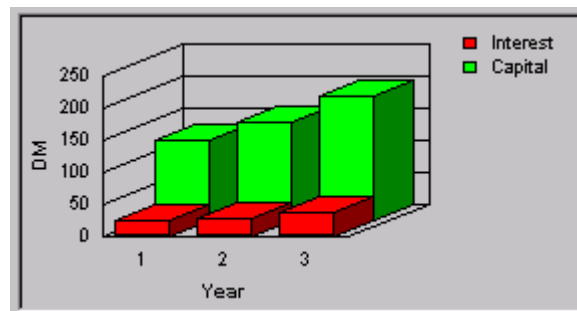
The graphic type vertical 3d bar shows each serie as a vertical 3d bar.

Example

Layout:



Runtime:

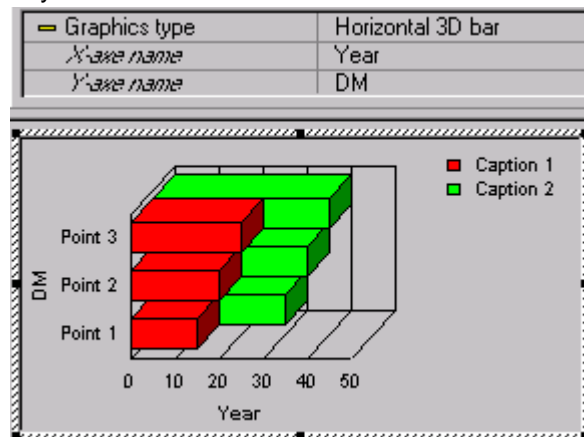


Horizontal 3D Bar

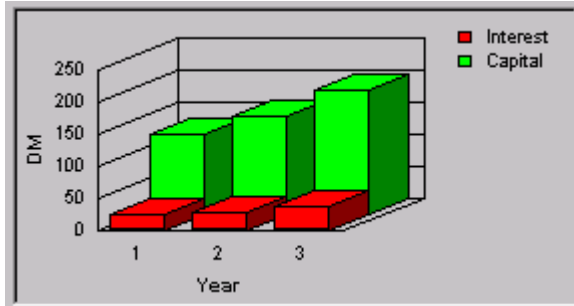
The graphic type horizontal 3d bar shows each serie as a horizontal 3d bar.

Example

Layout:



Runtime:



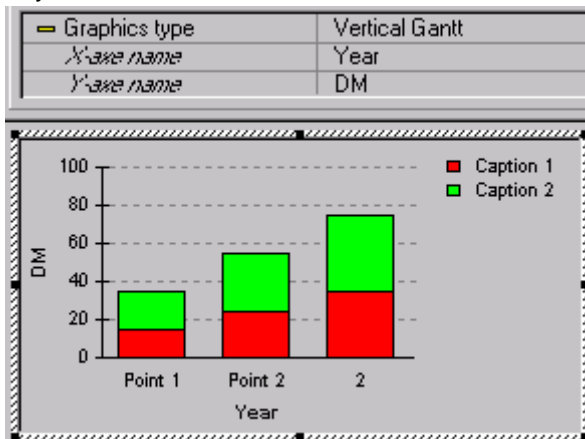
Vertical Gantt

The graphic type vertical gantt shows:

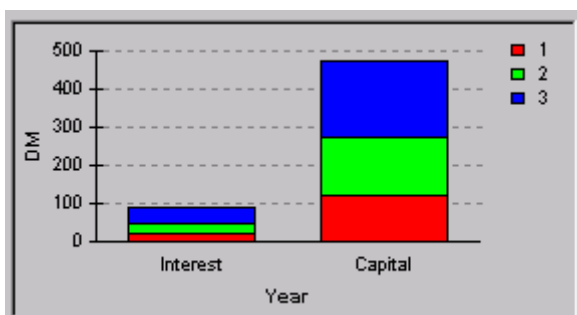
- Each serie as a single 1-dimensional vertical column.
- Each subset as a separate color.

Example

Layout:



Runtime:



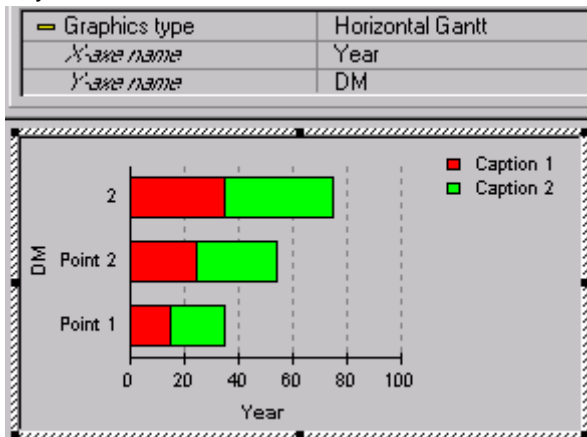
Horizontal Gantt

The graphic type horizontal gantt shows:

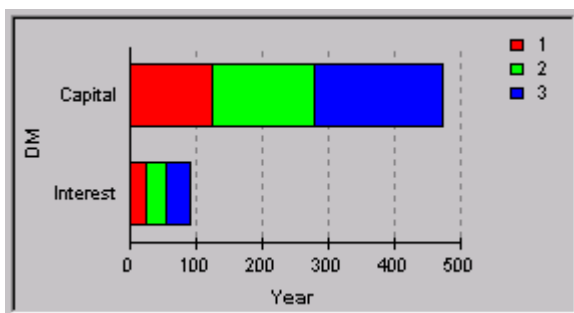
- Each serie as a single 1-dimensional horizontal column.
- Each subset as a separate color.

Example

Layout:



Runtime:



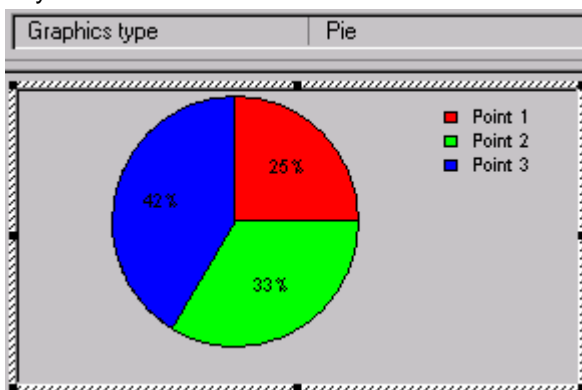
Pie

The graphic type pie shows:

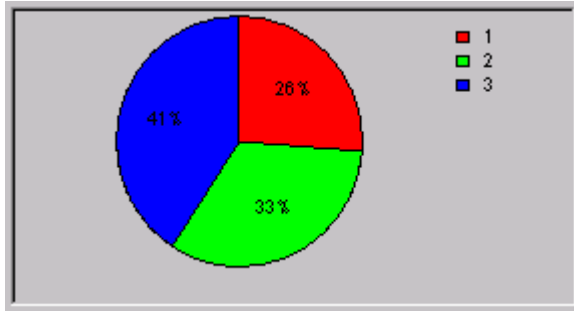
- All series for a subset combined.
- Each subset as a piece of a pie (with the size of the pie in percentage).

Example

Layout:



Runtime:



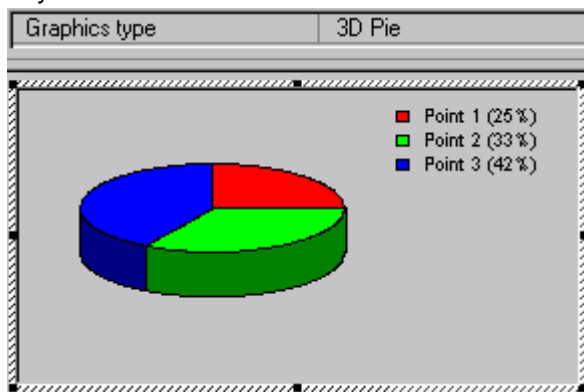
3D Pie

The graphic type pie shows:

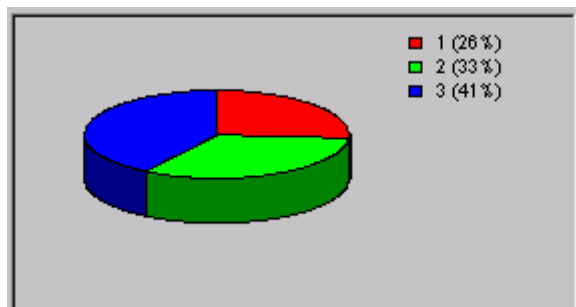
- All series for a subset combined.
- Each subset as a piece of a 3d pie (with the size of the pie in percentage).

Example

Layout:



Runtime:



Add / delete layout components

Designer provides a wide variety of options for adding and deleting the following types of layout components:

- Application header/footer
- Workarea
- Workarea header/footer
- Page
- Element

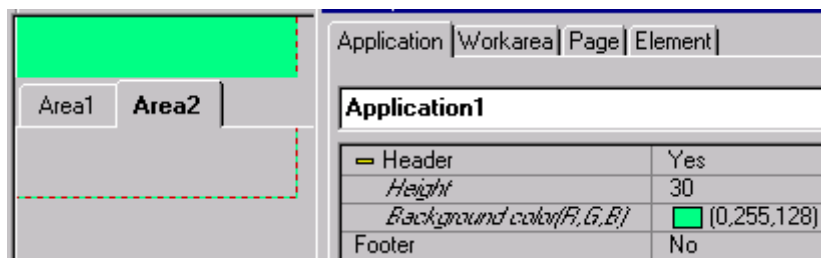
- Group of elements

Application header/footer add/delete

Application header

To add the application header:

1. Set application property <Style> = <Notebook>.
2. Set application property <Header> = <Yes>.
3. Set application header properties <Height> and <Background color(R,G,B)>.



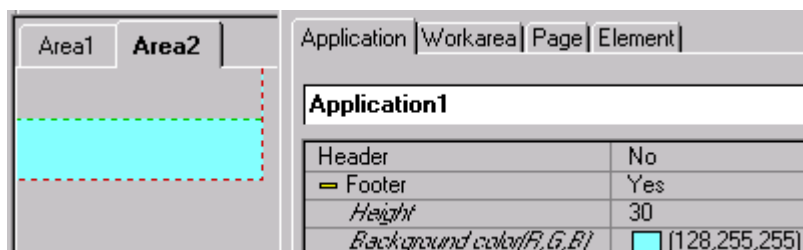
To delete the application header:

1. Set application property <Header> = <No>.

Application footer

To add the application footer:

1. Set application property <Style> = <Notebook>.
2. Set application property <Footer> = <Yes>.
3. Set application footer properties <Height> and <Background color(R,G,B)>.



To delete the application footer:

1. Set application property <Footer> = <No>.

Workarea insert / shift / remove

- Insert second workarea (switch to application <Style> = <Notebook>)
- Insert (before / after existing workarea)
- Shift (left / right)
- Remove
- Remove all but 1 workarea (switch to application <Style> = <Simple>)

Application: Switch to notebook style

Notebook style means multiple work areas, each with a tab.
When switching to notebook style:

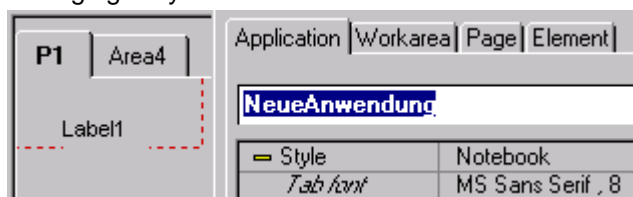
- The current layout becomes the first workarea.
- A second (empty) workarea is added.

Example

Assume the following layout with <Style> = <Simple>.



Changing <Style> to <Notebook> results in the following:



Workarea insert before / after

A new workarea is created by inserting before or after an existing workarea.

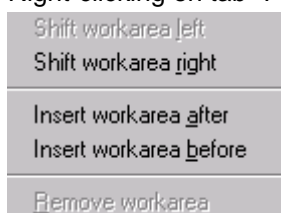
Note: Workareas can only be inserted if the Application property <Style> = <Notebook>.

Example

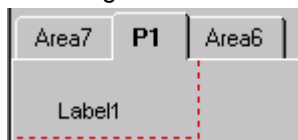
Assume the following layout.



Right-clicking on tab "P1" results in the following context menu:



Selecting "Insert workarea before" results in the following:



Workarea shift left / right

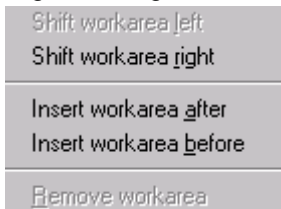
A workarea can be shifted to the left or right.

Example

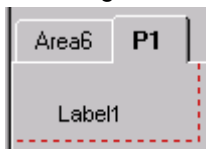
Assume the following layout.



Right-clicking on tab "P1" results in the following context menu:



Selecting "Shift workarea right" results in the following:



Workarea remove

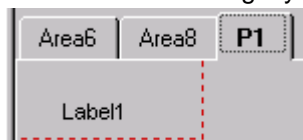
A workarea can be removed.

Note: Workareas can only be removed if the Application property <Style> = <Notebook> and at least 3 workareas exist.

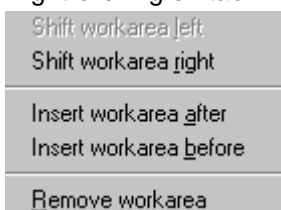
Note: To delete a workarea when only 2 workareas exist: Change the Application <Style> to <Simple>.

Example

Assume the following layout.



Right-clicking on tab "Area6" results in the following context menu:



Selecting "Remove workarea" and clicking "Yes" in the resulting dialog results in the following:



Application: Switch to simple style

Simple style means a single work area (with no tab).

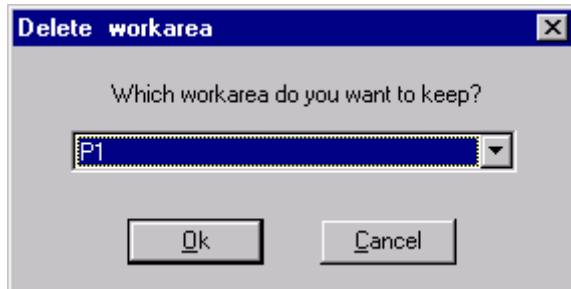
When switching to simple style, a dialog appears in which the single workarea that will remain in the layout is selected.

Example

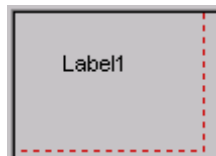
Assume the following layout with <Style> = <Notebook>.



Changing <Style> to <Simple> causes the following dialog to appear:



Select the workarea to keep results in the following:

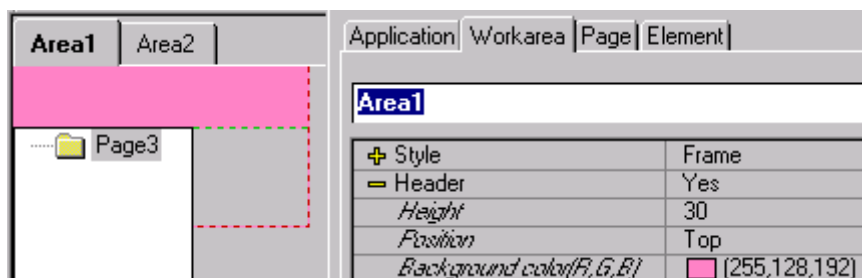


Workarea header/footer add/delete

Workarea header

To add the workarea header:

1. Set workarea property <Style> = <Frame>.
2. Set workarea property <Header> = <Yes>.
3. Set workarea header properties <Height>, <Position> and <Background color(R,G,B)>.



To delete the workarea header:

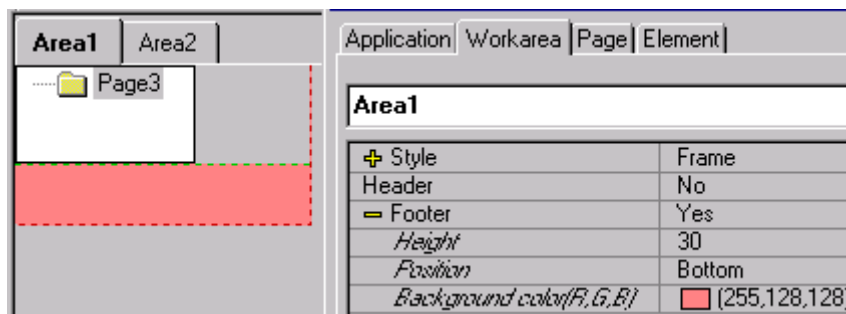
1. Set workarea property <Header> = <No>.

Workarea footer

To add the workarea footer:

1. Set workarea property <Style> = <Frame>.
2. Set workarea property <Footer> = <Yes>.

3. Set workarea footer properties <Height>, <Position> and <Background color(R,G,B)>.



To delete the workarea footer:

1. Set workarea property <Footer> = <No>.

Page insert / shift / remove

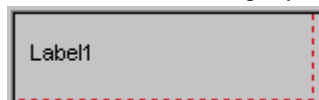
- Create a frame (switch to workarea <Style> = <Frame>) (required for multiple pages)
- Page insert below / above
- Page insert as subnode
- Page shift up / down
- Page remove
- Delete the frame (switch to workarea <Style> = <Simple>) (leaves a single page)

Workarea switch to frame style

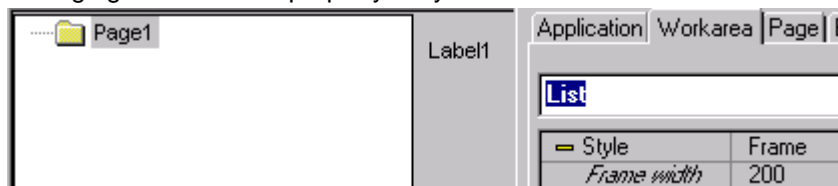
When changing the workarea property <Style> to <Frame>, the contents of the workarea become the contents of the single page in the workarea.

Example

Assume the following layout:



Changing the workarea property <Style> to <Frame> results in the following layout:



Resizing the frame results in the following:



Note that the size of the layout (the red dotted lines) includes the frame.

Page insert below / above

A new page is created by inserting below or above an existing page.

Note: Pages can only be inserted if the Workarea property <Style> = <Frame>.

Example

Assume the following layout.

Right-clicking on the icon/text "Page1" results in the following context menu:

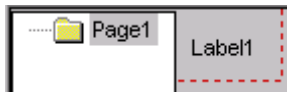
Selecting "Insert node above" ("node" = page) results in the following:

Page insert subnode (below)

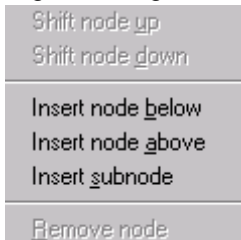
A new page can be created as a subpage of an existing page.

Example

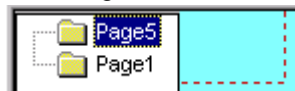
Assume the following layout.



Right-clicking on the icon/text "Page1" results in the following context menu:



Selecting "Insert subnode" ("node" = page) results in the following:

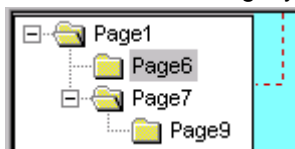


Page shift up / down

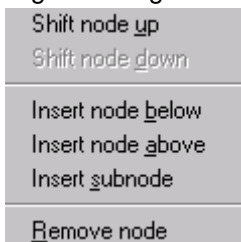
The location of a page in the frame can be shifted up or down in relation to other pages on the same level in the frame.

Example

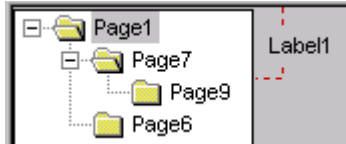
Assume the following layout.



Right-clicking on the icon/text "Page7" results in the following context menu:



Selecting "Shift node up" ("node" = page) results in the following:



Page remove

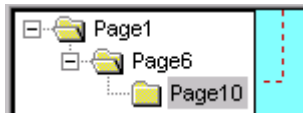
A page can be removed from the frame.

Note the following:

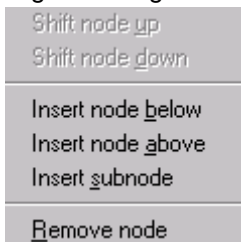
- The subnodes of a removed page are also removed.
- The last remaining top-level node can only be removed by changing the workarea property <Style> from <Frame> to <Simple>.

Example

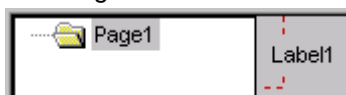
Assume the following layout.



Right-clicking on the icon/text "Page6" results in the following context menu:



Selecting "Remove node" ("node" = page) and clicking "Yes" in the resulting dialog results in the following:



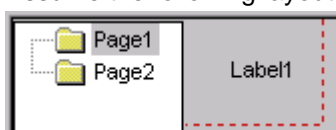
Right-clicking on the icon/text "Page1" results in a context menu with the option "Remove node" disabled.

Workarea switch to simple style

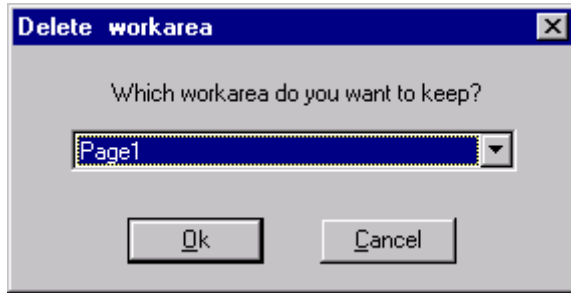
When changing the workarea property <Style> to <Simple>, all pages except for 1 will be deleted from the layout.

Example

Assume the following layout:



Changing the workarea property <Style> to <Simple> results in the following dialog:



Selecting the page ("workarea" in the above dialog) and clicking "OK" results in the following:



Note that the size of the layout (the red dotted lines) included the frame.

Element add / delete

Most of the standard windows functions such as copy / cut / paste / delete are supported in the Designer for elements.

- Add
- Copy
- Cut
- Paste
- Delete

Add element

To add an element to the layout:

- Click on the element toolbar. The mouse cursor changes to cross-hairs.
- Click with the mouse on the page where the element is to be added.

Example

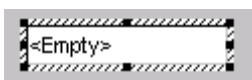
Click with the cursor the EntryField icon.



Move the cursor to the position on the page where the element should be added.



Click. The element is added.

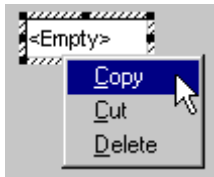


Copy element

An element can be copied (and pasted).

Example

Right click on the element. A context menu appears.



Click on Copy. The element has been copied to the clipboard.

Cut element

An element can be cut (to the clipboard).

Example

Right click on the element. A context menu appears.



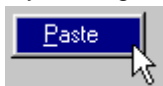
Click on Cut. The element is copied to the clipboard and deleted from the layout.

Paste element

An element can be pasted to the layout (from the clipboard).

Example

Assume an element has been copied to the clipboard using copy or cut. Position the cursor in the layout. Right click. A context menu (with typically only a single selection) appears.



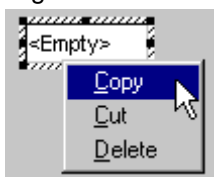
Click on Paste. The element is pasted from the clipboard to the layout.

Delete element

An element can be deleted from the layout.

Example

Right click on the element. A context menu appears.



Click on Delete. The element is deleted from the layout.

ElementGroup add / delete

Most of the standard windows functions such as copy / cut / paste / delete are supported in the Designer for element groups.

Designer also provides functionality for writing and reading element groups from files.

- Select
- Copy
- Write to file
- Cut
- Paste
- Read from file to clipboard
- Delete

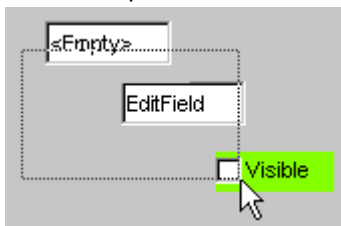
Select ElementGroup

A collection of elements can be selected as a group in any of the following ways:

- Hold down the left mouse button. Draw a box by moving the mouse. Any element totally or partially in the box become part of the group.
- Hold down the Ctrl key. Click on all elements that should be part of the group.

Example

This example shows how to select by drawing a box.

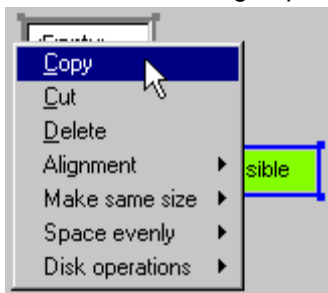


Copy ElementGroup

An element group can be copied (and pasted).

Example

Select the element group. Right click on any element in the element group. A context menu appears.



Click on Copy. The element group has been copied to the clipboard.

Write ElementGroup to file

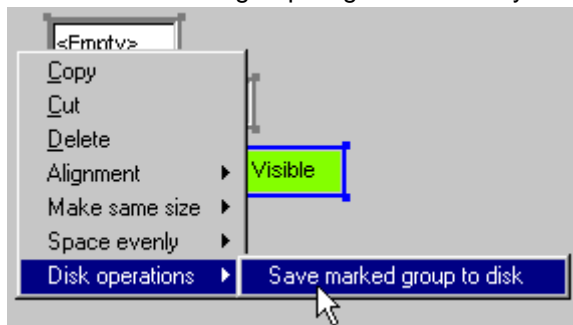
An element group can be written to a .vpg file.

See also:

- Menu item.
- Toolbar icon.

Example

Select the element group. Right click on any element in the element group. A context menu appears.



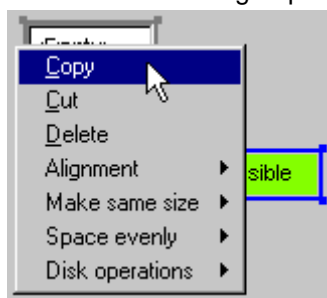
Click on Disk operations / Save marked group to disk. A dialog appears for entering the directory/filename of the .vpg in which the group will be saved.

Cut ElementGroup

An element group can be cut (copied to the clipboard and deleted from the layout).

Example

Select the element group. Right click on any element in the element group. A context menu appears.



Click on Cut. The element group is copied to the clipboard and deleted from the layout.

Paste ElementGroup

An element group can be pasted to the layout (from the clipboard).

Example

Assume an element group has been copied to the clipboard using copy or cut. Position the cursor in the layout. Right click. A context menu (with typically only a single selection) appears.



Click on Paste. The element group is pasted from the clipboard to the layout.

Read ElementGroup from file to clipboard

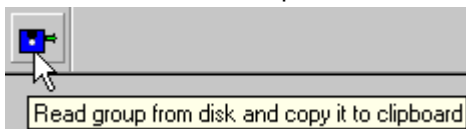
An element group can be copied from a .vpg file to the clipboard (an element group cannot be copied from a file to the layout directly).

See also:

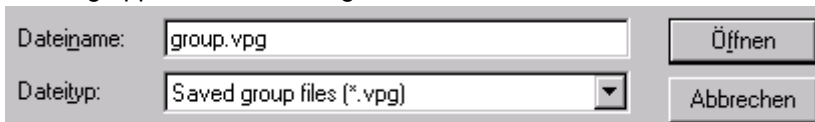
- Menu item.
- Toolbar icon.

Example

Click on the Read Group icon in the toolbar.



A dialog appears for selecting the file.



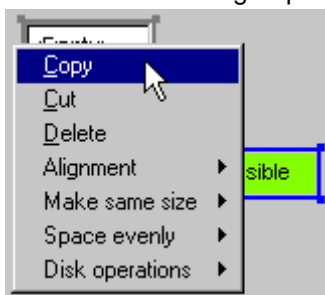
Click Open. The contents of the file have now been copied to the clipboard (and can now be added to the layout with Paste).

Delete ElementGroup

An element group can be deleted from the layout.

Example

Select the element group. Right click on any element in the element group. A context menu appears.



Click on Delete. The element group is deleted from the layout.

Properties

Properties have the following functions:

- Determine layout and components format.
- Specify external connections.

- Specify layout runtime functionality.

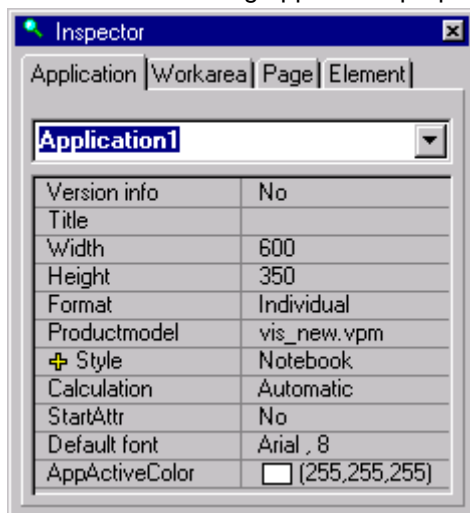
Component properties can be viewed and edited using the Inspector tool.

Layout and component format

Properties determine the format of the layout and the layout components.

Example

Assume the following application properties:



The properties that determine layout format are:

- Version info
- Title
- Width
- Height
- Format
- Style
- Default font
- AppActiveColor

Example

The following Label has property <Title> set to <Label text>.



External connections

The layout can have connections to the following products:

- The Application property <Productmodel> specifies the run-time model that is connected to the layout. The most important external product is the run-time model. The run-time model performs many functions, but most importantly performs all calculations. The layout properties can also specify data connections between model and layout.
- Media files can be included in the layout using the property <Media file>.

- DLL.
- Help file.

Layout runtime functionality

Properties can specify runtime layout functionality.

Only a very limited runtime functionality without an external connection is supported within Designer.

Runtime functionality for the following external connections is very simple, in that the external program is simply called from the Designer (and after that Designer has no effect on the connection):

- Media file
- DLL
- Help file

Designer provides a wide range of functionality for controlling the run-time model connection during runtime.

Run-time functionality without external connection

Only a very limited runtime functionality without an external connection is supported within Designer.

This functionality is dynamic during runtime, but no external connection (ie, model, dll, etc.) is required.

This includes:

- Visualize / Devisualize (dynamic visibility of workareas / pages).
- EntryField minimum / maximum values.

Run-time model connection

Designer properties provide a wide range of functionality for controlling the run-time model connection during runtime. This includes, for example (this list may not be exhaustive), control of the following functionality:

- Data input / selection to the model attributes (controlled by layout).
- Computation in the model (controlled by layout).
- Add / delete array members (controlled by layout).

Data connections model/layout

The following model data sources are available for layout components:

- Model attributes.
- Model attribute properties.
- Model properties.

Model attributes

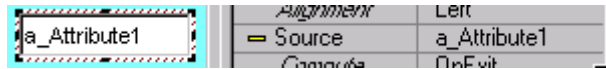
A layout component can display the value of a model attribute.

Example

Assume that the model has the following attribute:



To display and modify this model attribute in the layout, the following EntryField could be added to the layout:



Model attribute properties

A model attribute can have properties that are also displayed in the layout:

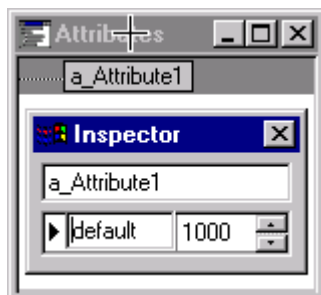
- Default
- Label
- Table

Default

The attribute property <default> can be used to recalculate the value of a property.

Example

Assume that the model has the following attribute with the following property <default>:



Note: The default property can also be defined by an equation.

The value of the attribute can using various mechanisms be re-set to the value defined by the property <default>.

Label

The text for a label in a layout can be defined not only in the layout, but also by the model attribute property <label>.

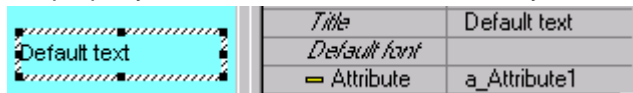
Example

Assume that the model has the following attribute with the following property <label>:



Note: The label property can also be defined by an equation.

The text for a label in the layout could then be defined by the attribute property <label> by specifying the property <Attribute> for the label in the layout.

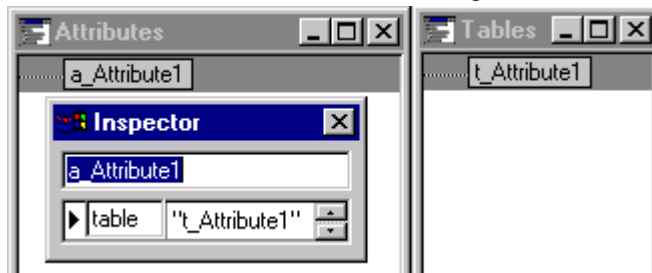


Table

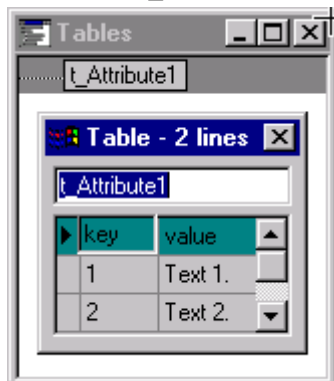
The index and text for a drop-down list, radio button group, etc., can be defined in a table in the model.

Example

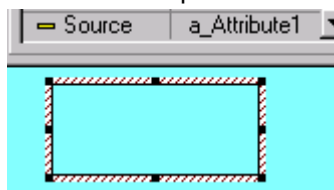
Assume that the model has the following attribute with the following property <table>:



The table <t_Attribute1> contains the following:



If a RadioGroup is added to the layout:



Then the runtime layout would contain the following:

○ Text 1.

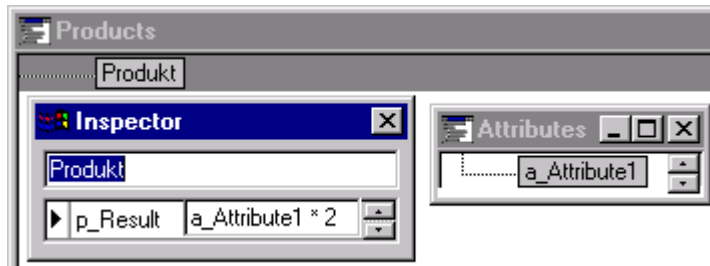
○ Text 2.

Model properties

Model properties (not attribute properties) are typically displayed in the layout as results.

Example

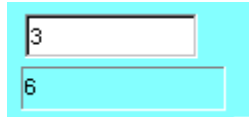
Assume that the model contains the property `<p_Result>` which is defined as shown in the following diagram:



Assume that the layout contains a ResultField:



Then the ResultField for `<p_Result>` would display the result in the runtime layout:



Data types

The following data types are supported by Designer.

- Single-line text
- Multi-line text
- Boolean
- Number
- Currency
- Date
- Arrays.

Text (single-line)

Text is the most common data type.

Integer values can also be displayed as text.

Multi-line Text

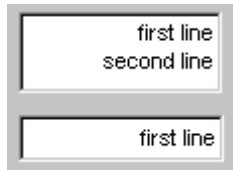
Multiline text can be entered in a Text EditField with `<Multiline> = <Yes>`. Multiline text contains special characters for line feed and carriage return that cannot be displayed in a single-line text EditField.

Example

Assume the following layout with a multiline and single-line text EditField.



In the runtime layout, only the first line of multiline text will be displayed in the single-line text EditField.



Boolean

Boolean data types are typically the values used by checkboxes.

Example

Assume that the layout has a CheckBox with property `<Source> = <a_Boolean1>`.



The only values that `<a_Boolean1>` could have would be 0 (CheckBox not checked) or 1 (CheckBox checked).

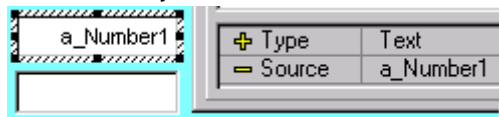
Number

The data type number is used for displaying numerical values with a fixed number of places after the decimal point.

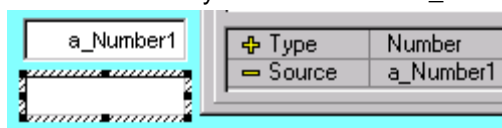
Example

Assume that the layout has 2 EntryFields.

The first EntryField Source is `<a_Number1>` with Type `<Text>`.



The second EntryField Source is `<a_Number1>` with Type `<Number>` (and `<Decimal position> = 2`).



Entering 123,456 (note that more than only 2 places are allowed after the decimal point) in the Text EntryField would result in the following:

123,456
123,45

Currency

The data type Currency is similar to Number. However Currency can also displayed a currency symbol.

Example

Assume that the layout has 2 EntryFields.

The first EntryField Source is <a_Number1> with Type <Text>.

a_Number1	Type	Text
	Source	a_Number1

The second EntryField Source is <a_Number1> with Type <Currency> (and <Decimal position> = 2).

a_Number1	Type	Currency
DM	Source	a_Number1

Entering 123,456 in the Text EntryField would result in the following:

123,456
123,45 DM

Date

The data type Date displays dates.

Example

Assume that the layout has 2 EntryFields.

The first EntryField Source is <a_Number1> with Type <Text>.

a_Number1	Type	Text
	Source	a_Number1

The second EntryField Source is <a_Number1> with Type <Date> (and <Format> = yy/mm/dd).

a_Number1	Type	Date
	Format	yy/mm/dd
	Alignment	Right
	Source	a_Number1

Entering 01.02.2345 (1 Feb 2345) in the Text EntryField would result in the following:

01.02.2345
45/02/01

Mask

A mask is a custom data type.

To provide possibility to have an edit field behavior different from the standard edit types (text, number, currency, date), a masking mechanism for edit fields should be implemented. It should be implemented as a new type (mask edit) supported for edits, result fields and grid columns and allowing the following mask symbols.

- # - must be a digit.
- N - must be a digit. If not entered will be set to '0'.
- & - must be a digit or system decimal delimiter.
- X - must be a letter.
- C - must be a letter. If not entered will be set to <space>.
- U - must be a letter. When entered converted to uppercase.
- L - must be a letter. When entered converted to lowercase.
- ? - any character may be entered.
- \ - the next character will be counted as a delimiter, not the mask character.

At the same time a separate formatting mechanism is necessary for date edits, date result fields and date grid columns to allow application specific date formats independent from the system settings. That can be achieved with introducing an additional Format property for date edits. The following patterns are allowed for this property.

- dd - must be a day of month (two digit, 1-31);
- mm - must be a month number (1-12);
- M, MM, MMM - must be a month name (English or defined by the system settings);
- yy - must be a year

To provide possibility to have an edit field behavior different from the standard edit types (text, number, currency, date), a masking mechanism for edit fields should be implemented. It should be implemented as a new type (mask edit) supported for edits, result fields and grid columns and allowing the following mask symbols.

- # - must be a digit.
- N - must be a digit. If not entered will be set to '0'.
- & - must be a digit or system decimal delimiter.
- X - must be a letter.
- C - must be a letter. If not entered will be set to <space>.
- U - must be a letter. When entered converted to uppercase.
- L - must be a letter. When entered converted to lowercase.
- ? - any character may be entered.
- \ - the next character will be counted as a delimiter, not the mask character.

At the same time a separate formatting mechanism is necessary for date edits, date result fields and date grid columns to allow application specific date formats independent from the system settings. That can be achieved with introducing an additional Format property for date edits. The following patterns are allowed for this property.

- dd - must be a day of month (two digit, 1-31);
- mm - must be a month number (1-12);
- M, MM, MMM - must be a month name (English or defined by the system settings);
- yy - must be a year (two digit, the current century is assumed);
- yyyy - must be a year (four digit);
- other characters considered as delimiters.

Entering / selecting data

Designer provides components for:

- Entering data (values for model properties and attributes)
- Selecting data (from model tables)

Designer also provides the following functionality for entering/selecting data:

- Element order. Determines the order that layout components will obtain the focus.
- Start attribute. The first attribute on start-up in the layout.
- Parameter search. Designer can automatically give the focus to layout components for entering required data.
- Data indicator. The data indicator indicates when all required data has been entered.

Entering data

Designer provides the following components for entering values:

- EditField. Allows entry of text, number, date, currency.
- ExtendedGrid. An EditField can be used within the ExtendedGrid to enter data.

Entering data: EditField

The EditField is used to enter text or numerical (decimal numbers, dates, currency amounts) values.

Entering data: ExtendedGrid

A column in an ExtendedGrid can use an EntryField for entering values.

Selecting data

Designer provides the following components for selecting data from a list of available values:

- CheckBox and RadioButton allow a boolean value to be selected.
- ComboBox and RadioGroup allows a value to be selected from a list of available values.
- ExtendedGrid can include a ComboBox in a column.

CheckBox

A CheckBox allows a value of 1 or 0 to be entered for a model attribute.

RadioButton

A RadioButton allows a value of 1 or 0 to be entered for a model attribute.

ComboBox

A ComboBox allows an integer value to be entered for a model attribute.

The integer value is selected by selecting a text item from the drop-down list provided by the ComboBox. The contents of the drop-down list and the integer associated with each text item is provided by a model table.

RadioGroup

A RadioGroup allows an integer value to be entered for a model attribute.

The integer value is selected by selecting a text item from the RadioGroup. The contents of the RadioGroup and the integer associated with each text item is provided by a model table.

ExtendedGrid

An ExtendedGrid column can use a ComboBox to select data.

Concept: Element order

The element order is the order in which the elements on a page will obtain the focus when using the tab key.

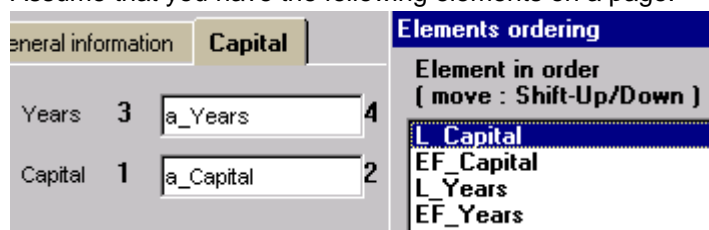
There is a menu item for displaying/hiding element order in layout.

The element order can be changed using the Elements (re)ordering dialog.

Note: Changing the element order is not possible if the page sequence default is set to standard.

Example

Assume that you have the following elements on a page.



You want a_Years to be before a_Capital. Click on EF_Years in the Elements ordering dialog and using the Shift-Up key combination move EF_Years above EF_Capital. Clicking OK results in the following:



Start attribute

The Start attribute is the first attribute that obtains the focus when the layout is first displayed.

The Start attribute can be:

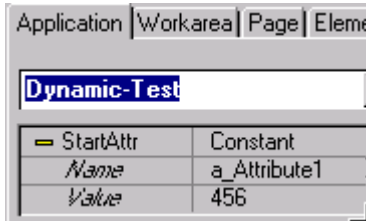
- Constant.
- Dynamic.

Start attribute: Constant

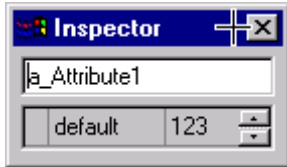
A Constant start attribute specifies the attribute and the value for the attribute.

Example

Assume that the start attribute is Constant and the attribute is a_Attribute1 with value 456.



a_Attribute1 property <default> = 123.



When the layout is started, the value for a_Attribute1 = 456.

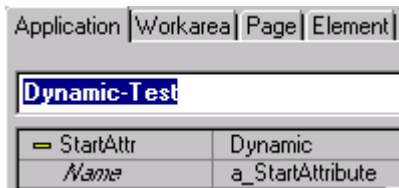


Start attribute: Dynamic

A Dynamic start attribute is an attribute that is specified by an attribute.

Example

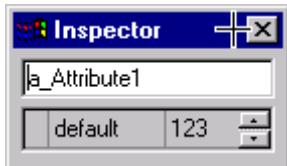
Assume that the start attribute is Dynamic and the attribute that specifies the dynamic start attribute = <a_StartAttribute>.



a_StartAttribute property <default> specifies the start attribute (<default> could include an equation).



a_Attribute1 property <default> = 123.



Parameter search

Assume that a model attribute is required to compute a model property (for example: p_Result = a_Variable1 * ...).

If a value for the attribute is entered in the layout AND the model property is displayed on the same page in the layout: The Designer enters parameter search mode for the remaining attributes required to calculate the model property.

Parameter search mode algorithm

In parameter search mode:

- Each remaining page can be opened 1 time (regardless of whether or not the page contains an entry element for a required attribute).
- If a required attribute is not entered on an open page and an attempt is made to open another page: parameter search mode ends.
- If the last required attribute is entered: parameter search mode ends.
- Attempting to open a page a 2nd time (the page contained no required data or the required data was entered the 1st time the page was opened) will cause a previously unselected page that contains required data to be selected.
- Attempting to open a page a 3rd time will cause parameter search mode to end.

If parameter search mode was started and ended:

- If a required attribute is entered on any page (not necessarily the page where the result is displayed): Parameter search model is restarted. Note the following:
 - Data already entered for required attributes is not part of the restarted parameter search.
 - Any page that contains no entry field for required data can be opened 1 more time.

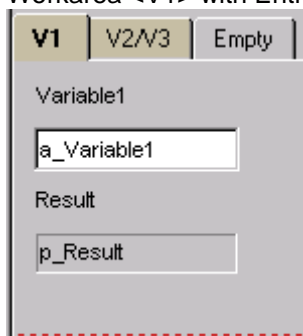
Example

The following example assume that the model contains property `p_Result` where:

- $p_Result = a_Variable1 * a_Variable2 * a_Variable3$

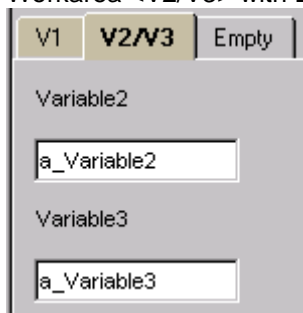
The example also assumes that the layout has the following structure:

- Workarea <V1> with EntryField for `a_Variable1` and ResultField for `p_Result`.



The screenshot shows a workarea with three tabs: 'V1', 'V2/V3', and 'Empty'. The 'V1' tab is selected. Below the tabs, there is a section labeled 'Variable1' containing an entry field with the text 'a_Variable1'. Below that is a section labeled 'Result' containing a result field with the text 'p_Result'. A red dashed border highlights the 'Variable1' and 'Result' sections.

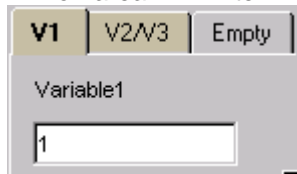
- Workarea <V2/V3> with EntryFields for `a_Variable2`, `a_Variable3`.



The screenshot shows a workarea with three tabs: 'V1', 'V2/V3', and 'Empty'. The 'V2/V3' tab is selected. Below the tabs, there is a section labeled 'Variable2' containing an entry field with the text 'a_Variable2'. Below that is a section labeled 'Variable3' containing an entry field with the text 'a_Variable3'. A red dashed border highlights the 'Variable2' and 'Variable3' sections.

- Workarea <Empty> with no components.

In workarea V1: Enter 1 for Variable1. Parameter search mode is started.



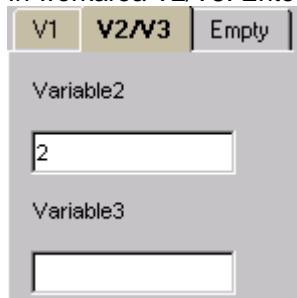
Select workarea Empty. Note that Empty can be selected 1 time, even though it contains no entry fields for required attributes.

Attempt to select workarea V1. Workarea V2/V3 is automatically selected.

Select workarea V1 or Empty. Both have previously been selected and data entry is required in neither. Therefore, parameter search mode has ended.

Select any workarea. Note that parameter search mode is not active.

In workarea V2/V3: Enter 2 for Variable2.



Select workarea Empty.

Attempt to select workarea V1. Note that workarea V2/V3 is selected.

Enter 3 for Variable3. Note that parameter search mode has ended since all required attributes were entered.

Data indicator

A data indicator is used to indicate that all required data has been entered.

Specifying default path location of DI bitmaps.

Purpose of the "data indicator" (DI) is to visualize, if data exists/has been entered in logical parts of the application. Especially in voluminous or highly structured applications with more than only a few areas/pages, it is very helpful for the user to see, which part of the application contains data and which part does no

Restrictions of the data indicator

Although it would be very useful, not only to visualize the existence of data but also to visualize and represent their logical status ("either the data is consistent and error free or the data violates validation rules"), this cannot be supplied by the data indicator

To obtain and especially to keep consistency of this kind of context specific information would require, that, if any status relevant information in any page is modified, the status indicator of all other pages have to be checked and updated. This can consequently only be achieved by checking (validating) all concerned input information as well as by computing all result information. The performance overload caused by this behavior or concept would be significantly.

Moreover, not all status information can be obtained by checking input definitions – a significant amount of errors and validation violation messages are not "attribute-bound" in the application model but defined in conjunction with the computational logic of results (in workbench terminology: computed properties in the product tree). Errors caused in this way cannot be mapped directly to pages in the application, therefore no actualization of the DI can be done in consequence.

Due to this reasons the DI does not represent the validation status of data.

Represented conditions by the data indicator

The DI is a page specific flag, that represents for the user of the application one of the following logical conditions for the page:

- Condition: Data exists. The page contains data the DI is "ON".
- Condition: Data does not exist. The page contains no data the DI is OFF.

So far as it is very difficult to find an appropriate way for defining of the DI state using visual element states of the page, a rule defining the DI state must exist in the computational model for every page needed the DI. This rule should be connected to an attribute property of the computational model having two possible values 0 and 1.

DI symbol file format

So far as the DI symbol set should be editable the following approach is used. In the Designer a special directory (as a part of the Designer installation) containing the DI bitmaps is supported. All the DI bitmaps from this directory are displayed in the secondary dialog for DI symbol selection in the inspector. A certain standard set of the DI bitmaps is provided with the Designer. If a new DI symbol is necessary, any bitmap of this set may be edited with some graphical tool (ex. MS Paint) or a new bitmap may be added to the set. The DI bitmaps used in XPL-application are saved in the byte code of the application (VPC-file) in a special resource section. This approach provides flexibility in defining of various DI symbols and allows to distribute XPL-applications without any additional files containing DI symbols bitmaps (also acceptable for Intranet/Internet platform).

Displaying data

Designer provides the following components for displaying data:

- EntryField can be used to display data.
- ResultField is the standard component for displaying results.
- Label (dynamic) can be used to display results in a textual format.
- Grid / ExtendedGrid can include EntryFields and ResultFields in the columns.
- Graphic (dynamic) can display arrays of data from a Grid or ExtendedGrid in several different visual formats.

Displaying data: EditField

The EntryField can be used to display resultant data in the following ways:

- Default property for an EntryField can specify a result.
- Reformatted Entry. An EntryField can reformat entered data.

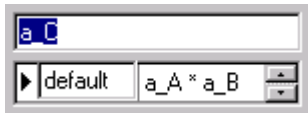
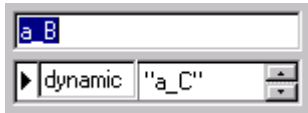
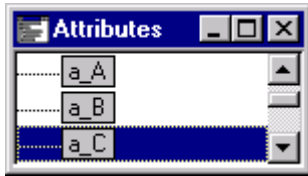
Default

The model attribute property <default> can specify an equation. This default value will be recalculated and displayed in the layout if:

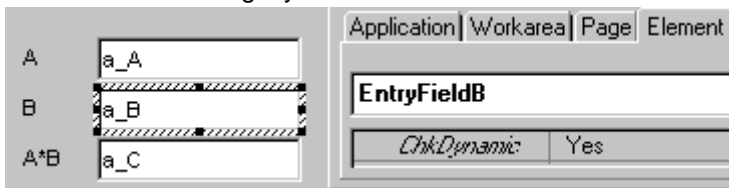
- The second attribute has property <Dynamic> = <(name of first attribute)>.
- The second attribute has property <ChkDynamic> = <Yes>.
- The second attribute is changed.

Example

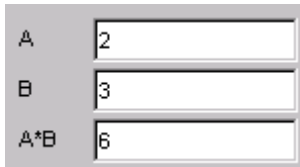
Assume the following model:



Assume the following layout:



The runtime layout would then display $a_A * a_B$ in the EntryField a_C .



Reformatted Entry

An EntryField can reformat entered data.

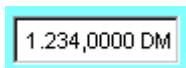
Example

Assume that an EntryField has <Type> = Currency and <Decimal position> = 4.

Entering the following



would result in the following display



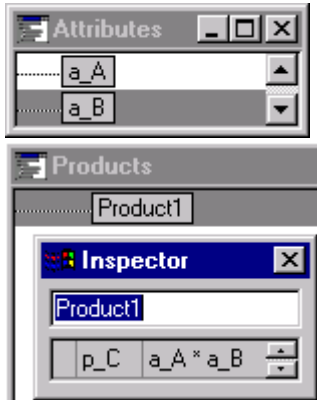
ResultField

The ResultField is the standard component for displaying results. The ResultField normally displays a property of a product in the model.

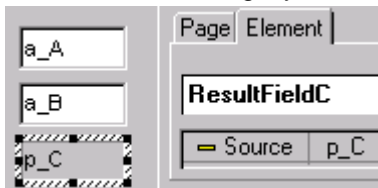
Note: The drop-down list for layout property <Source> does not display product properties from the model (only attributes are displayed). The property name must always be typed in.

Example

Assume the following model.



Assume the following layout.



The runtime layout would then be:



Dynamic Label / Border

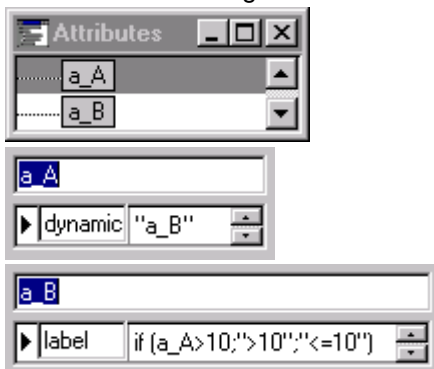
Dynamic Labels, Borders, Lines, Multimedia elements can be used to display results.

To create a dynamic label:

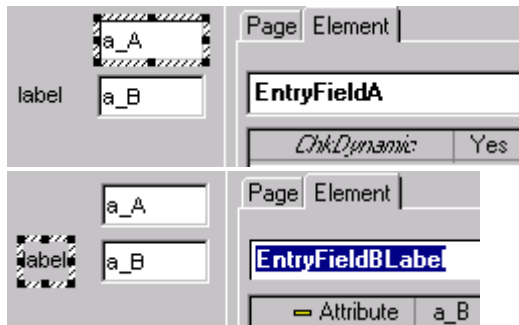
- A model attribute must have property <label> defined. The definition normally uses an "If" statement with a string return value.
- A second model attribute has attribute <dynamic> = <"first attribute">.
- The layout element for the second model element has property <dynamic> = <Yes>.

Example

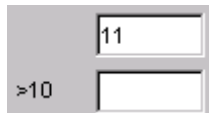
Assume the following model:



Assume the following layout:



The runtime layout would then be:



Grid / ExtendedGrid

The Grid and ExtendedGrid can be used to display results.

Dynamic Graphic

A dynamic graphic can be used to display results.

Compute result property

After attributes have been changed, the resultant properties must be recalculated. The recalculation can occur based on the following:

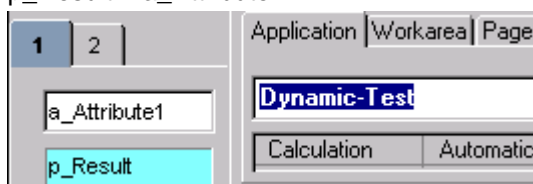
- Change page.
- Pushbutton is pressed.
- Enter data.
- Change focus. (a different layout component is clicked on).

Change page

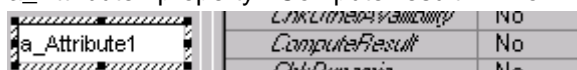
The results on a page are recalculated when the page changes if property <Calculation> = <Automatic>.

Example

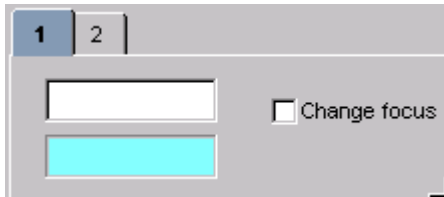
Assume that the layout contains EntryField for a_Attribute1 and ResultField for p_Result where $p_Result = a_Attribute1 * 2$.



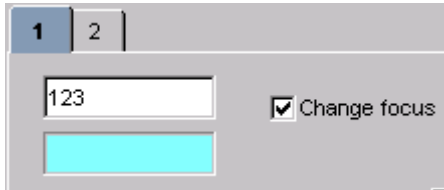
a_Attribute1 property <ComputeResult> = No.



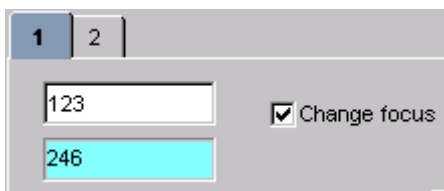
Open the layout. Note that the EntryField for a_Attribute1 is empty.



If data is entered in the EntryField and the CheckBox is checked, the result is still not re-calculated.



If workarea 2 is selected and then workarea 1 reselected, the result is re-calculated.



Pushbutton

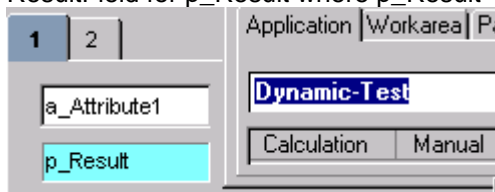
The results on a page can be recalculated when a PushButton is pressed if:

- Application property <Calculation> = <Manual>.
- PushButton property <Action> = <Compute>.

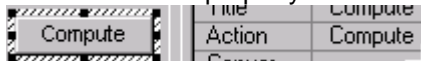
Example

Assume the following layout:

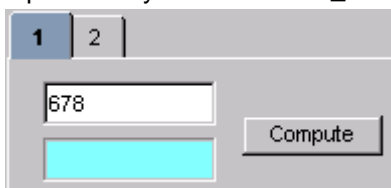
- Calculation = Automatic
- EntryField for a_Attribute1
- ResultField for p_Result where $p_Result = a_Attribute1 * 2$



- PushButton with property <Action> = <Compute>.

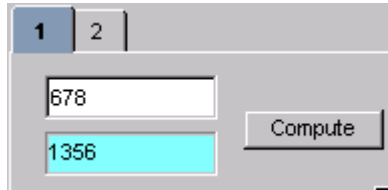


Open the layout. Enter for a_Attribute1 value 678.



Change pages. Note that the result is not calculated.

Click the Compute PushButton. The result is calculated.



Enter data

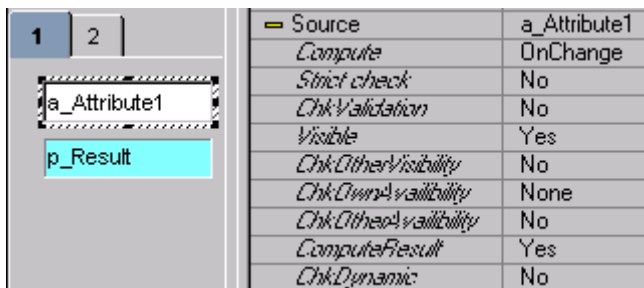
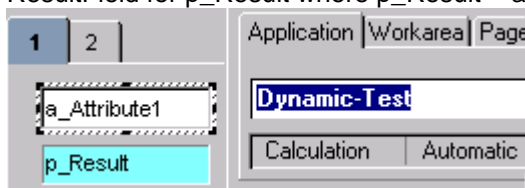
The results on a page can be recalculated when data (a single letter or number) is entered if:

- Application property <Calculation> = <Automatic>.
- EntryField property <ComputeResult> = <Yes>.
- EntryField property <Compute> = <OnChange>.

Example

Assume the following layout:

- Calculation = Automatic
- EntryField for a_Attribute1 with:
 - <ComputeResult> = <Yes>.
 - <Compute> = <OnChange>.
- ResultField for p_Result where $p_Result = a_Attribute1 * 2$



Open the layout. Click in EntryField for a_Attribute1 and then press 3. Note that the result is computed immediately.



Press 4. The result of 68 is computed immediately.

Change focus

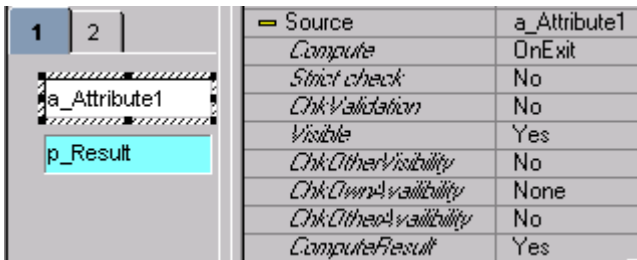
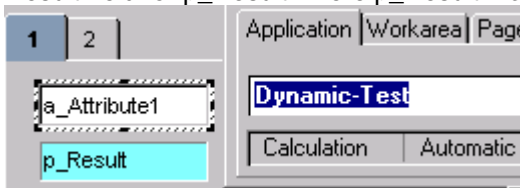
The results on a page can be recalculated when the focus changes if:

- Application property <Calculation> = <Automatic>.
- EntryField property <ComputeResult> = <Yes>.
- EntryField property <Compute> = <OnExit>.

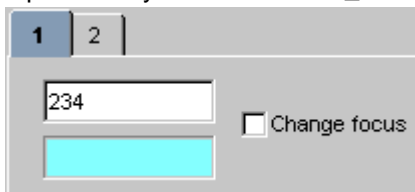
Example

Assume the following layout:

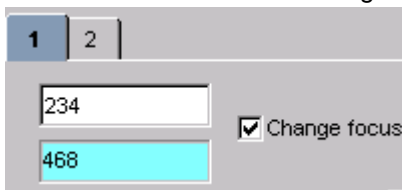
- Calculation = Automatic
- EntryField for a_Attribute1 with:
 - <ComputeResult> = <Yes>.
 - <Compute> = <OnExit>.
- ResultField for p_Result where $p_Result = a_Attribute1 * 2$



Open the layout. Enter 234 a_Attribute1. Note that the result is not computed.



Click on the CheckBox <Change focus>. The result is re-calculated.



Re-compute attribute default/label

The following can be re-computed:

- Attribute property <default>
- Attribute property <label>

when the following occurs:

- Enter data.
- Change focus.

Enter data

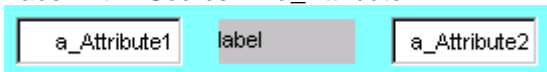
The default and label for Attribute2 can be recalculated when data (a single letter or number) is entered in an EntryField for Attribute1 if:

- Application property <Calculation> = <Automatic>.
- EntryField for Attribute1 property <ComputeResult> = <Yes>.
- EntryField for Attribute1 property <Compute> = <OnChange>.
- Model attribute Attribute2 property <default> = (calculation).
- Model attribute Attribute2 property <label> = (calculation).

Example

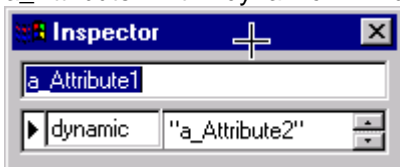
Assume the following layout:

- Calculation = Automatic
- EntryField for a_Attribute1 with:
 - <ComputeResult> = <Yes>.
 - <Compute> = <OnChange>.
- EntryField with <Source> = a_Attribute2.
- Label with <Source> = a_Attribute2.

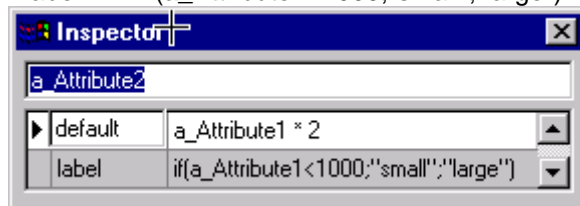


Assume the following model for the above layout:

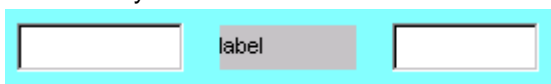
- a_Attribute1 with <dynamic> = <"a_Attribute2">



- a_Attribute2 with:
 - <default> = <a_Attribute1 * 2>
 - <label> = <if(a_Attribute1<1000;"small";"large")>



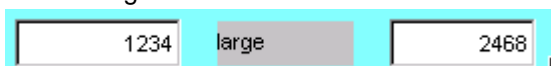
Start the layout.



Click in the a_Attribute1 EntryField. Click on 1. The a_Attribute2 Label and EntryField are updated.



Enter for a_Attribute1 1234. Note that a_Attribute2 Label and EntryField are updated with each entered digit.



Change focus

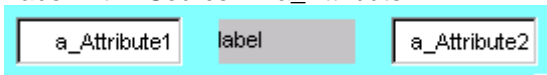
The default and label for Attribute2 can be recalculated when the focus changes (a different component or page is selected) if:

- Application property <Calculation> = <Automatic>.
- EntryField for Attribute1 property <ComputeResult> = <Yes>.
- EntryField for Attribute1 property <Compute> = <OnExit>.
- Model attribute Attribute2 property <default> = (calculation).
- Model attribute Attribute2 property <label> = (calculation).

Example

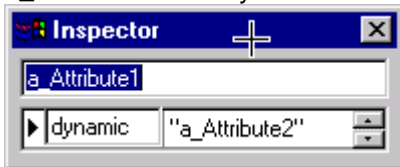
Assume the following layout:

- Calculation = Automatic
- EntryField for a_Attribute1 with:
 - <ComputeResult> = <Yes>.
 - <Compute> = <OnExit>.
- EntryField with <Source> = a_Attribute2.
- Label with <Source> = a_Attribute2.

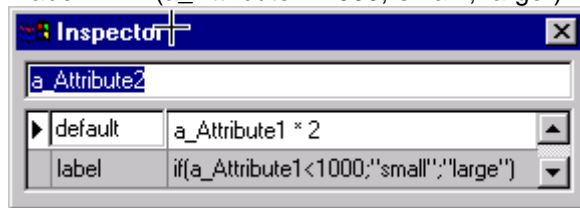


Assume the following model for the above layout:

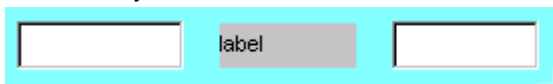
- a_Attribute1 with <dynamic> = <"a_Attribute2">



- a_Attribute2 with:
 - <default> = <a_Attribute1 * 2>
 - <label> = <if(a_Attribute1<1000;"small";"large")>



Start the layout.

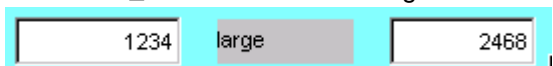


Enter 1 in the a_Attribute1 EntryField.

Change the focus by clicking on a different component. The a_Attribute2 Label and EntryField are updated.



Enter for a_Attribute1 1234. Change the focus. Note the a_Attribute2 Label and EntryField contents.



Visibility

The visibility of layout components can be defined in:

- Design-time layout
- Runtime layout

Design-time visibility (element levels)

Designer provides a level functionality for complex layouts or layouts where multiple components occupy the same space in a page:

- 8 levels exist in a layout: Level 0 ... Level 7.
- An element can be assigned to any level.
- An element is assigned to Level 0 by default.
- For Levels 1...7: All of the elements on a level can be hidden by deselecting that level.

This is normally used for overlaying elements that are optionally displayed.

Example:

Assume that a layout has a Label and EntryField (with Level specified as 1) as shown below.



From the main menu: Select Options / Levels / Level 1 to deselect Level 1. Note that the EntryField disappears:



Now the EntryField a_Months could be added at the same location in the layout with Level 2 specified. In the runtime layout either the a_Years or the a_Months EntryField would be displayed.

Runtime layout visibility

The visibility of layouts components in the runtime layout can be:

- Static (the visibility does not change in the runtime model)
- Dynamic (the visibility can change in the runtime model)

Static visibility

A workarea, page, or element can be made non-visible by setting the property <Visible> = <No>.

Example

Assume the following layout.



In the runtime layout the EntryField would not be visible.



Dynamic visibility

Dynamic visibility means that the visibility of the component can change during runtime. The following components can have dynamic visibility:

- Workareas, Pages, Elements (with property `<Visible> = <Property>`)
- EntryField (property `<Check>`)
- Workareas/Pages (with `<Visualize>` and `<Devisualize>`)

Dynamic visibility with `<Visible> = <Property>`

Workarea, page, and element visibility can dynamically change in the runtime layout with the following:

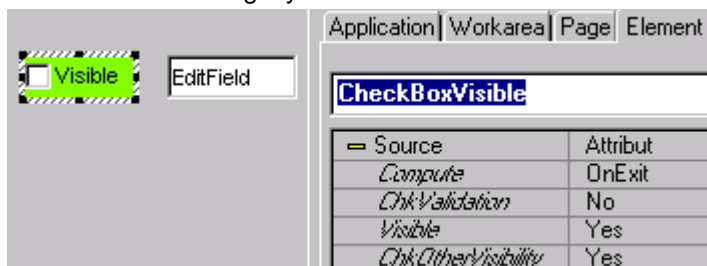
- An boolean-value element (typically a CheckBox) is used to set the value of an attribute for the visibility (the visibility can be controlled by a model property or function as well).
- The property `<ChkOtherVisibility>` of the above element is set to `<Yes>`.
- The element property `Source . Visible = . <Property>` = the value of the attribute for visibility.

See also:

- `ChkOtherVisibility`: Setting the default value.

Example

Assume the following layout:

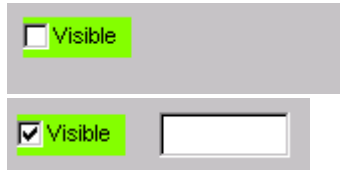


Note above:

- `<Source> = <Attribut>`
- `<ChkOtherVisibility> = <Yes>`



In the runtime layout:



Dynamic visibility: EntryField property <Check>

The EntryField supports an added mechanism for dynamic visibility: The property <Check>. To use this property for dynamic visibility:

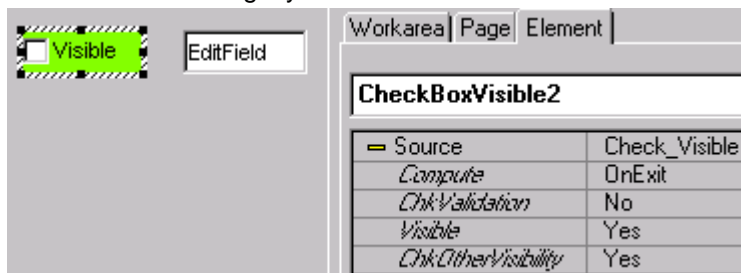
- The element property Source . Visible = <Check>
- In the model: The Attribute which is the Source for the EditField has property <visible> which returns a boolean (1 or 0).
- The property <ChkOtherVisibility> of any element that might cause the model property value to change is set to <Yes>.

See also:

- ChkOtherVisibility: Setting the default value.

Example

Assume the following layout:



Note above:

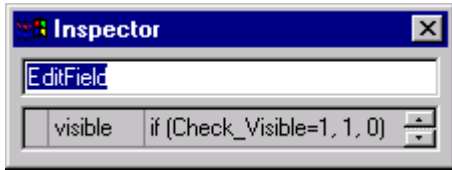
- <Source> = <Check_Visible>
- <ChkOtherVisibility> = <Yes>



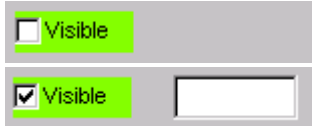
Note above:

- <Source> = <EditField>
- <Visible> = <Check>

In the model:



In the runtime layout:



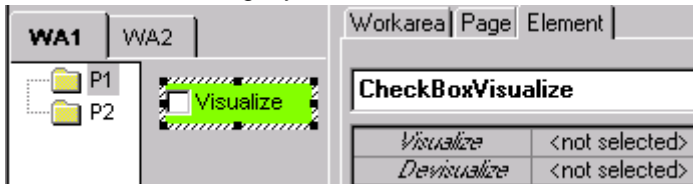
Dynamic visibility using <Visualize> and <Devisualize>

Dynamic visibility of workareas and pages can be implemented using <Visualize> and <Devisualize>. Typically this is implemented with a CheckBox (or similar element) with property <Visualize> and/or <Devisualize> with the required workareas or pages specified.

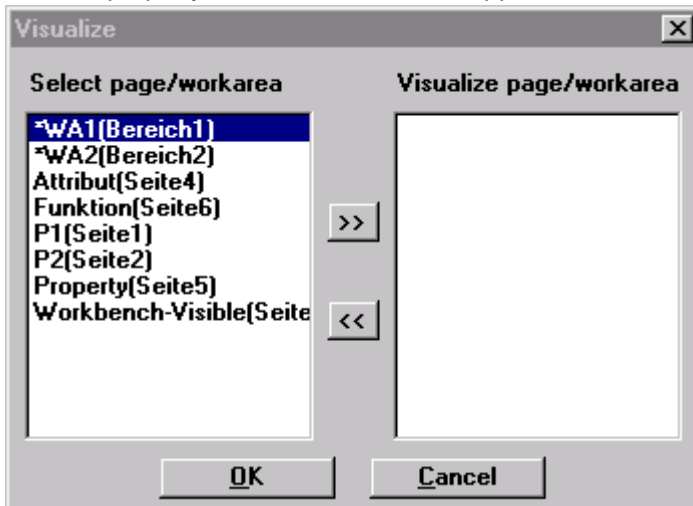
Example

Note: This example demonstrates property <Visualize>. Property <Devisualize> is the opposite of <Visualize>.

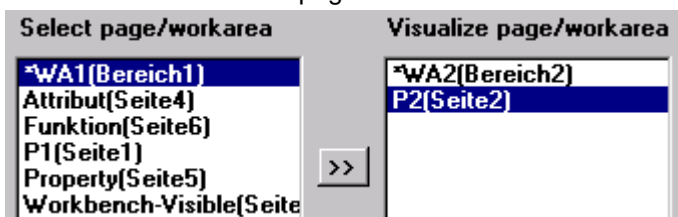
Assume the following layout:



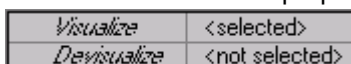
Click on property <Visualize>. A button appears. Click on the button and the following dialog appears:



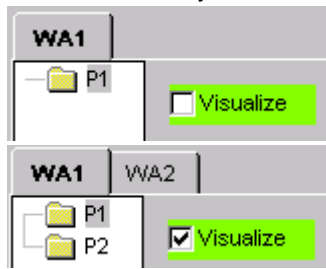
Select the workarea and page for visualization as shown below:



Click OK. Note that the property <Visualize> = <selected>.



In the runtime layout



Availability

An element is not available if the element is grey and input is not possible.

"Availability" in Designer depends on whether or not the element is required for computing any result in a specified list of results.

Element availability can be:

- Static. The list of elements required for computing each result in the result list does not change.
- Dynamic. For 1 or more results in the list of results: The list of elements required to compute the result is dynamic (using the "If" expression).

Results list

"Availability" in Designer for an element is determined by whether or not the model attribute for the element is required for a list of model properties (results) specified for the element.

The following types of result lists can be specified:

- No result (no list is specified (the element is always available))
- All results (in the layout)
- Element result list (a list for the specific element)
- Page result list (a list for the page the element is on)

See also:

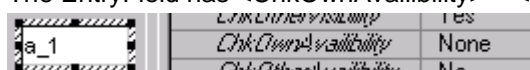
- Setting the default value for the results list.
- Property ChkOwnAvailability (specifies the result list).

Result list: None

If no result list is specified: The element is always available.

Example

The EntryField has <ChkOwnAvailability> = <None>:



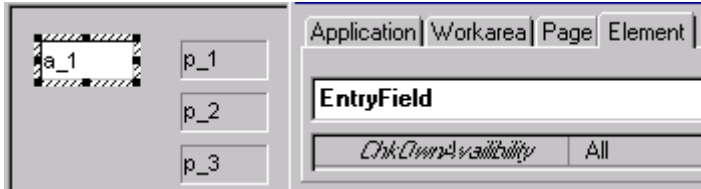
The EntryField is always available.

Result list: All

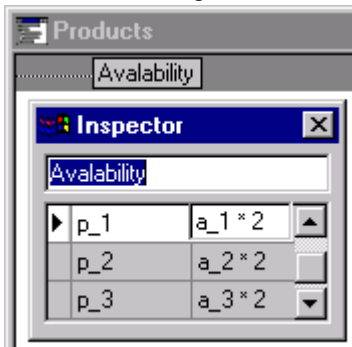
If element property <ChkOwnAvailability> = <All>: The element is available if the element is required for computing any model result property (on any page) in the layout.

Example

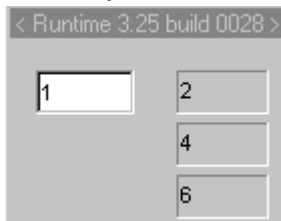
Assume the following layout



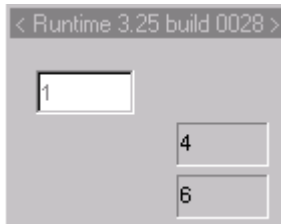
with the following model



The EntryField would be available in the runtime-layout:



However, if the ResultField for p_1 was removed, the EntryField would be unavailable:

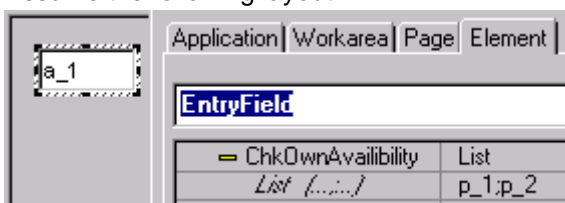


Result list: Element

If element property <ChkOwnAvailability> = <List>: The element is available if the element is required for computing any model result property in the list specified by element property <List (...;...)>.

Example

Assume the following layout



with the following model



The EntryField would be available in the runtime-layout since a_1 is required for computing p_1.

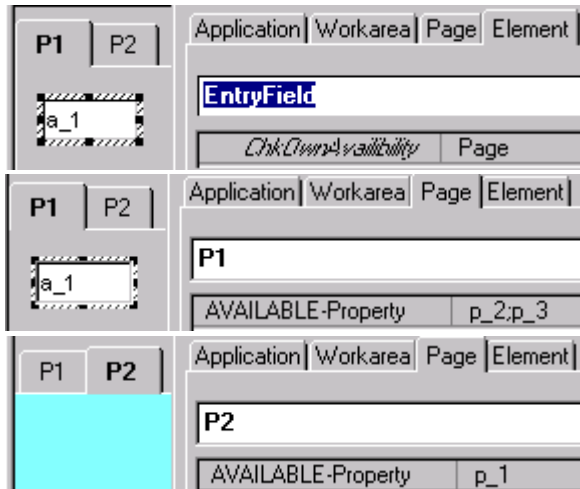
Result list: Page

If element property <ChkOwnAvailability> = <Pages>: The element is available if the element is required for computing any model result property in the list specified by the page property <AVAILABLE-Property> for the page that contains the element.

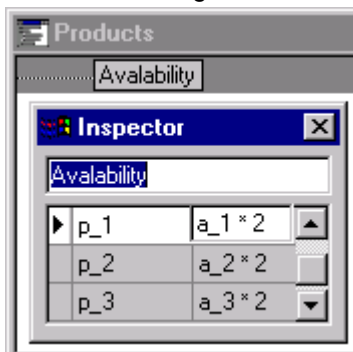
Examples

Example 1

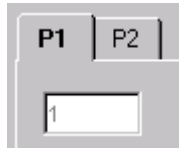
Assume the following layout



with the following model

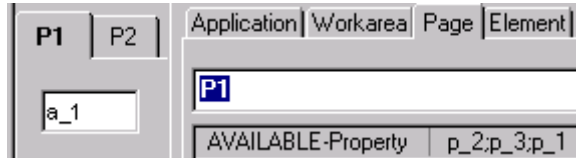


The EntryField would not be available in the runtime-layout since p_1 is not specified for P1 property <AVAILABLE-Property>.

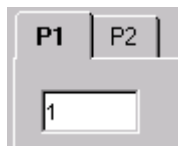


Example 2

Assume the above layout, except that P1 <AVAILABLE-Property> includes p_1:



The EntryField would be available in the runtime-layout since p_1 is specified for P1 property <AVAILABLE-Property>.



Static Availability

"Static availability" means that list of elements required for computing each result in the result list does not change.

Dynamic availability

"Dynamic availability" means that the availability of an element can change during runtime. The following is true for dynamic availability:

- The list of elements required for computing each result in the result list does NOT change (as for static availability). The list can be ALL, LIST, or PAGE.
- The required elements for a property do change (using an IF expression).
- If changes to an element should possibly cause changes in the availability of any element in the layout: The element property <ChkOtherAvailability> = <Yes>.

See also:

- Setting the default value for ChkOtherAvailability.
- Property ChkOtherAvailability.

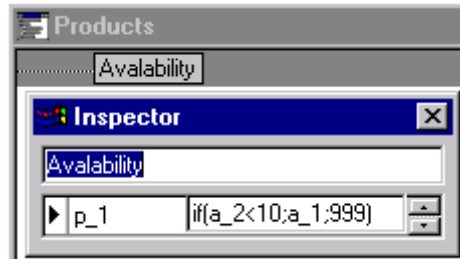
Example

Assume the following layout

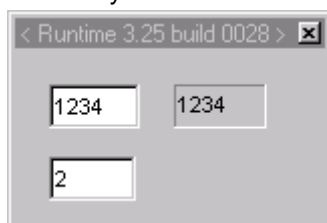




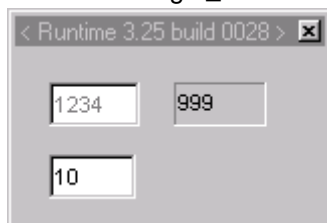
with the following model



The EntryField would be available if $a_2 < 10$. For example, in the following $a_2 = 2$:



In the following $a_2 = 10$ and the a_1 EntryField is unavailable (and the ResultField = 999):



Arrays

Designer provides a wide range of functionality for arrays. This functionality can be divided into 3 basic categories:

- Table
- Iteration
- Multiple inclusion

Table

Model components are required for defining the table.

Layout components are required for displaying the table.

The model property filter can be used to display only a subset of the table rows in the layout (rows in the table cannot be altered, added or deleted in the design-time or runtime layout).

Model components

The simplest form of a table requires the following in the model:

- A table.
- An attribute with property `<table>` that specifies the table.

Table

The table is defined in the Workbench subwindow "Tables".

The table must have the following 2 columns:

- Column 1: Key column. The key column contains a unique integer identifier (index) for each row in the table. When a row in the column is selected in the layout (for example via a ComboBox or RadioGroup), the value in the key column for that row is assigned to the attribute whose property <table> specifies the table.
- Column 2: Value column. The value column typically contains text. The rows of text in this column are displayed in the layout when selecting a row via a ComboBox or RadioGroup.

Attribute

The model attribute property <table> references the table and contains the key for the selected row.

Example

Attribute a_Table1 property <table> = <"t_Table1">:

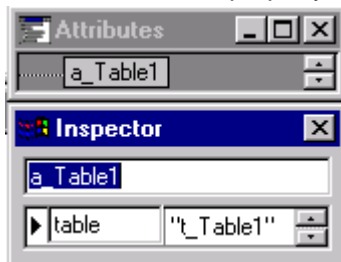
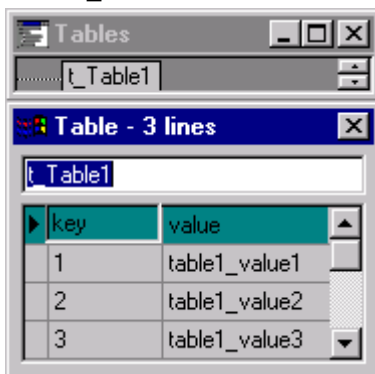


Table t_Table1 contains columns <key> and <value> with 3 rows of data:



See the next page for an example layout that uses the above example model.

Layout components

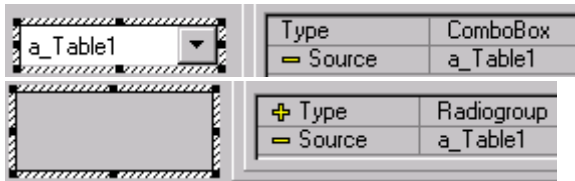
The following layout components are typically used to display the rows of table text and select a row:

- CheckBox
- RadioGroup

The component property <Source> = the model attribute whose property <table> specifies the model table.

Example

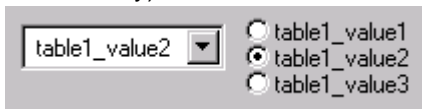
Assume the model described on the previous page.
 Assume the following layout:



In the runtime layout:



Selecting "table1_value2" from either layout element would cause the same selection to be displayed in the other element (note: element property <Compute> should be <OnChange> to see the change immediately).



Accessing key/value without ComboBox/RadioGroup

The previous 2 pages described how to use a ComboBox or RadioGroup to display the contents of a table and select the index.

This page describes how the key and value of the selected row can be accessed.

Key

The key of the selected row is stored in the model attribute whose property <table> specifies the table. This attribute is available throughout the model.

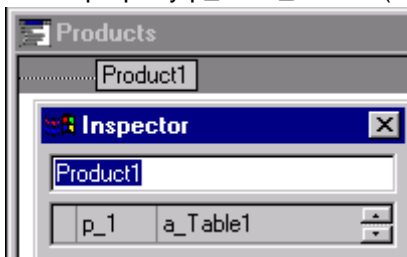
The attribute can be displayed in the layout by defining the value of a model property equal to the attribute.

Value

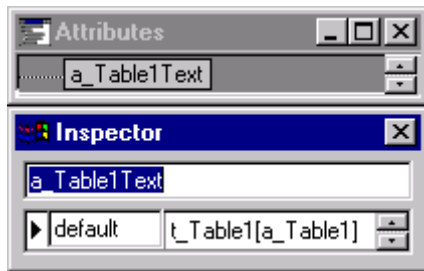
The value can be accessed via an equation of the format table_name[table_index].

Example

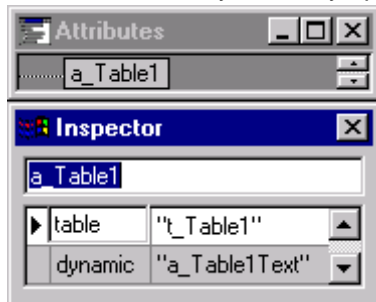
Model property p_1 = a_Table1 (attribute that holds the key of the selected row).



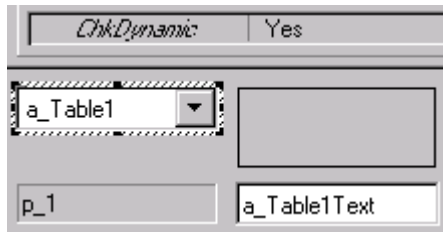
Model attribute a_Table1Text property <default> = value in table t_Table1 in the selected row.



a_Table1Text is dynamically updated by changes to a_Table1.



The layout has a ResultField for p_1 and an EditField for a_Table1Text. <ChkDynamic> = <Yes> for the ComboBox and the RadioGroup.



In the runtime layout, choosing a table row via the ComboBox or the RadioGroup would also display the key and value for the selected row:



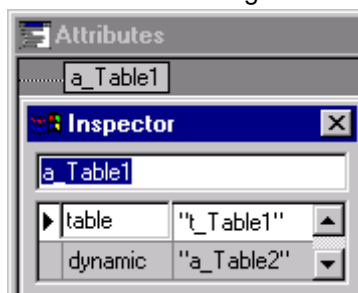
Table filter

Table rows cannot be added or deleted via the layout.

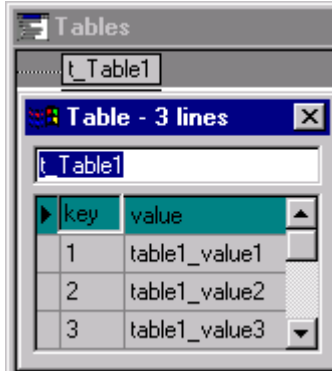
However, the rows displayed in the layout can be filtered as defined by the model attribute property <filter>.

Example

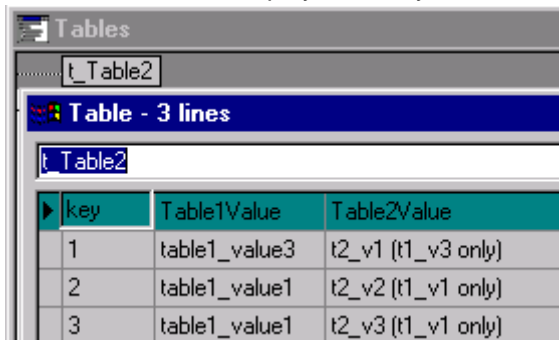
Assume the following model:



t_Table1:



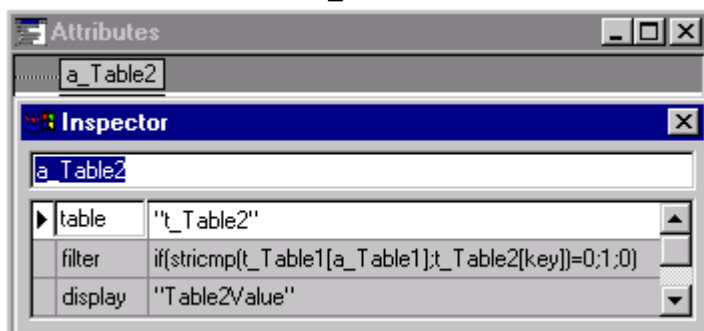
t_Table2 contains 3 columns. Column 2 contains the same values as column 2 in t_Table1. Column 3 contains values for display in the layout.



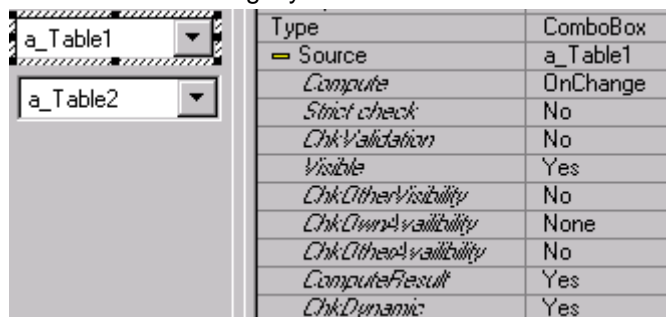
Attribute a_Table2 property <filter> =

`if(stricmp(t_Table1[a_Table1];t_Table2[key])=0;1;0)`

The above expression specifies that only rows in t_Table2 will be displayed in the layout if t_Table2 column 2 value = selected t_Table1 column 2 value.

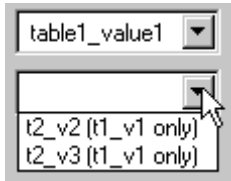


Assume the following layout with ComboBox's for both tables:

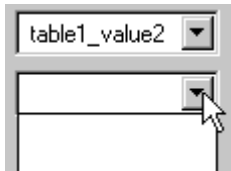


In the runtime layout:

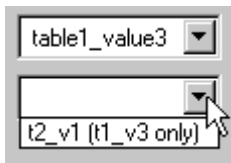
If t_Table1 row 1 is selected in the upper ComboBox, only t_Table2 rows 2 and 3 are available from the lower ComboBox.



If t_Table1 row 2 is selected in the upper ComboBox, no t_Table2 rows are available from the lower ComboBox.



If t_Table1 row 3 is selected in the upper ComboBox, only t_Table2 row 1 and 3 is available from the lower ComboBox.



Iteration (with Grid)

An array can be generated with an Iteration.
 Model components.
 Layout component consists of a Grid.

Model components

The simplest form of an iteration requires the following in the model:

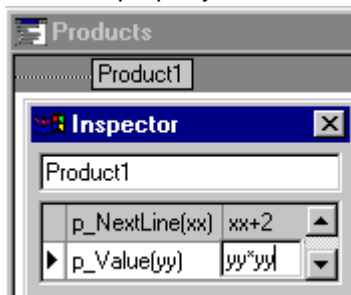
- An iteration attribute which specifies the total number of iterations.
- A start attribute which defines the number for the first iteration.
- A next-line property that generates the number for an iteration from the number of the previous iteration.
- A value for each iteration.

Example

Iteration and start attributes:



Next-line property and a value for each iteration:



Note: "xx" and "yy" used above could be changed to any string with no effect. See the next page for an example layout that uses the above example model.

Layout components

A grid is typically used to display all iterations.

Example

Assume the model described on the previous page.

The following 2 EditField's in the layout are used for entering the start and iteration attributes:



Note: The above elements are not absolutely required in the layout. For example, the start and iteration could be defined with attribute property <default> in the model.

Assume the following Grid:

Iteration	(next line)	Value
a_Start	p_NextLine()	p_Value()
[(next line)]-1	p_NextLine()	p_Value()
1	2	3

p_NextLine() and p_Value() in the model do not use a defined index (on the previous page "xx" and "yy" were used). The common index for all attributes/properties in the Grid is defined by specifying Grid property <Key column> (1 for this example). The computed value of each iteration will be stored in the iteration attribute specified by property <Attribute>:

Key column	1
Iteration type	Attribute
Attribute	a_Iteration

The computed value for the iteration is saved to the attribute specified for the Key column. Therefore key column property <Usage> = <Input>.

The Grid column whose <Property> specifies the property that generates the iteration is specified with <Next line> (in this example 2).

Note, however, that the key column in the first row must be defined somehow; this is accomplished by defining <Attribute> = <a_Start>.

Property for	Column 1
Title	Iteration
Width	70
Usage	Input
Attribute	a_Start
Next line	2

Column 2 <Usage> = <Output>, since this column "outputs" a value to other columns (column 1).

Property for	Column 2
Title	(next line)
Width	80
Usage	Output
Property	p_NextLine()

Finally, Column 3 contains the value:

Property for	Column 3
Title	Value
Width	70
Usage	Output
Property	p_Value()

In the runtime layout: Entering 7 for a_Start and 3 for a_Iteration would have the following results (p_Value(x) = x*x):

7	3	
Iteration	(next line)	Value
7	9	49
9	11	81
11	13	121

Multiple pages

Arrays of data can be generated using the multiple pages functionality in the Designer. Multiple pages can be implemented with

- No Grid
- Grid
- ExtendedGrid

Multiple pages without Grid

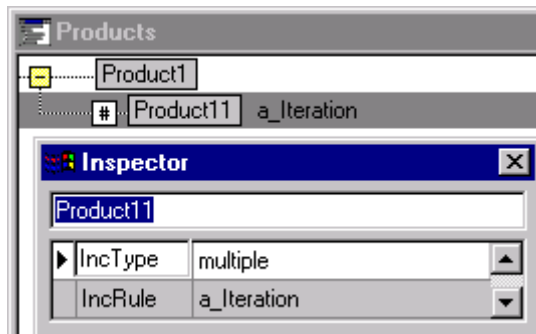
Model components.
 Layout components.
 Adding/deleting array elements.

Model components

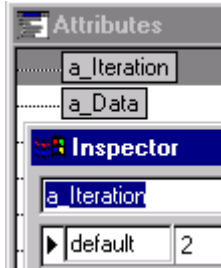
The model requires a product subnode with `<IncType> = <multiple>`.
 The subnode property `<IncRule> = iteration attribute`.
 Calculating a result for display in the main node requires an equation of the form:
 $p_Result = a_Attribute[\#iteration_attribute]$
 Including a result for each iteration on each multiple page requires in the subnode an equation of the form:
 $p_Result(zyx) = \text{if}(zyx=\#iteration_attribute;p_Result;0)$

Example

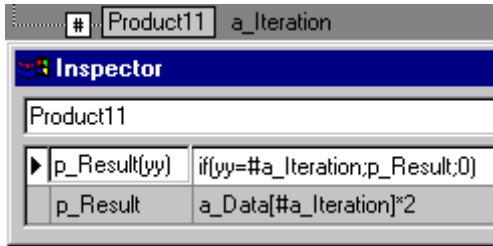
Product main node Product1 has a subnode Product11 with `<IncType> = <multiple>` and `<IncRule> = <a_Iteration>`.



2 default array iterations are defined with `a_Iteration <default> = 2`.
`a_Data` is the data attribute for each iteration.



p_Result(yy) ("yy" can be any text) is required for displaying the result for an iteration on the layout subnode for that iteration.



p_Result is used to sum all results in the Product11 subnodes on the Product1 node as p_Total.



See the next page for an example layout that uses the above example model.

Layout components

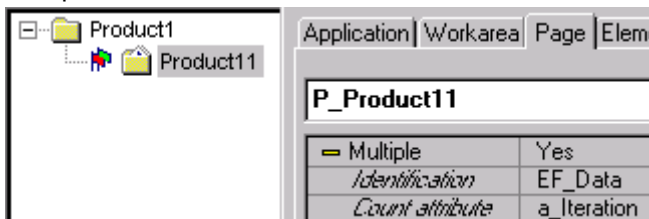
The layout requires a page with multiple subnodes.
The multiple subnode has:

- <Multiple> = <Yes>
- <Count attribute> = iteration attribute.

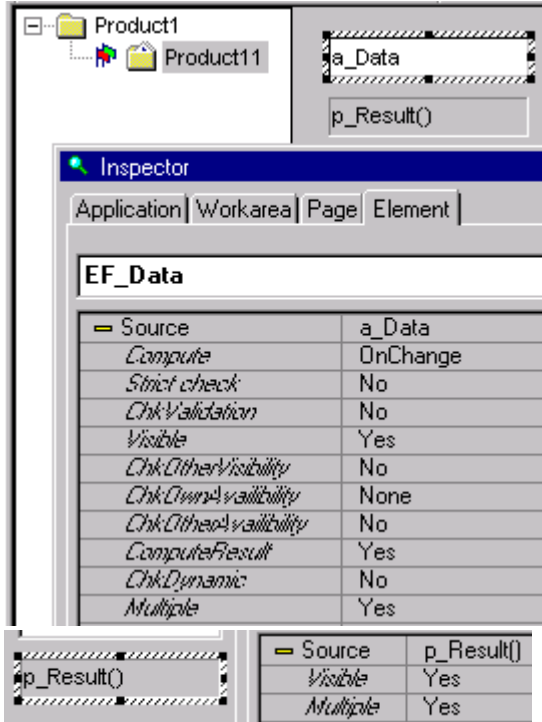
The layout also requires the elements to enter data and display results.

Example

Assume the model described on the previous page.
Assume the following layout:
Multiple subnode Product11 with <Count attribute> = <a_Iteration>.



On subnode Product11: An EditField for a_Data and a ResultField for p_Result(), both with <Multiple> = <Yes>.

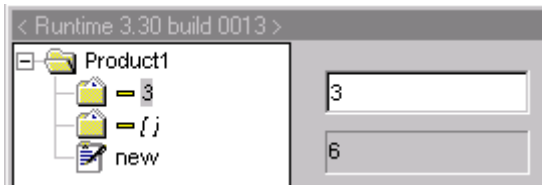


The main node has a ResultField for p_Total.



In the runtime layout:

A subnode:



The main node:



Add/delete array members

Addition and deletion of array members is implemented directly in the layout.

The subnode properties <Caption New> and <Caption Delete> define the text displayed in the runtime layout when the mouse is right-clicked in order to delete a subnode or when the icon is double-clicked to add a subnode.

Example

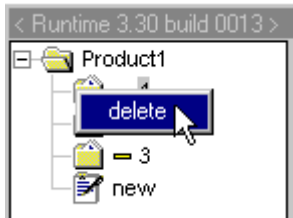
Assume the layout/model described on the previous pages with <Caption New> and <Caption Delete> as shown:



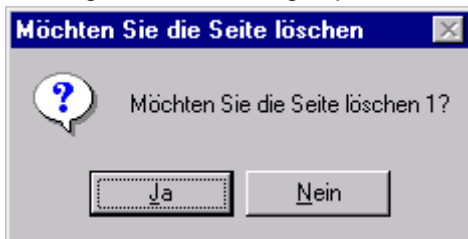
In the runtime layout, double-clicking on the "new" icon would create a new array member:



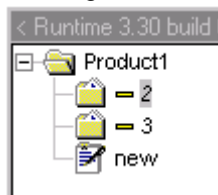
Right-clicking on a subpage icon would cause a popup menu with "delete" to appear:



Clicking on "delete" brings up the following dialog.



Clicking "Yes" deletes the array member.



Multiple pages with Grid

Model components.

Layout components.

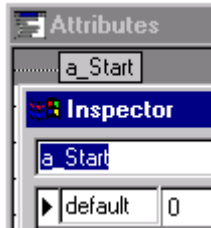
Model components

In addition to the requirements for multiple pages without a grid, with a grid the model requires the following:

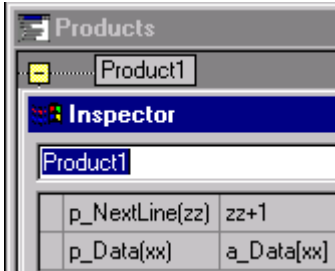
- Start attribute with default.
- Property that generates the index for the next iteration.
- Properties that reference the attributes to be displayed in the Grid.

Example

Start attribute a_Start with <default> = 0.



Upper node property p_NextLine() generates the index for the next iteration.
 Upper node property p_Data() references attribute a_Data.



See the next page for an example layout that uses the above example model.

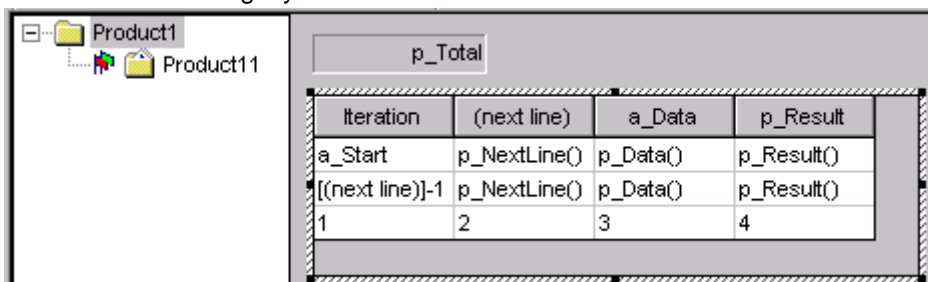
Layout components

In addition to the requirements for model subnodes, the layout requires the following definitions for the Grid:

- # of columnsn
- Key column
- Iteration attribute
- Start attribute
- Next line column and property
- Data input column and property (note: The data input column is specified as an "output" column).
- Result column and property.

Example

Assume the model described on the previous page.
 Assume the following layout:



<Column> = 4 (number of columns).
 Column 1 is the key column.
 Iteration attribute is a_Iteration.

Type	Grid
Column	4
Columns moveable	Yes
Key column	1
Iteration type	Attribute
Attribute	a_Iteration

Start attribute is a_Start.

Column 2 generates the next line iteration.

Property for	Column 1
Title	Iteration
Width	70
Usage	Input
<i>Attribute</i>	a_Start
<i>Next line</i>	2
Property for	Column 2
Title	(next line)
Width	70
Usage	Output
<i>Property</i>	p_NextLine()

Column 3 is the output column for inputting a_Data value.

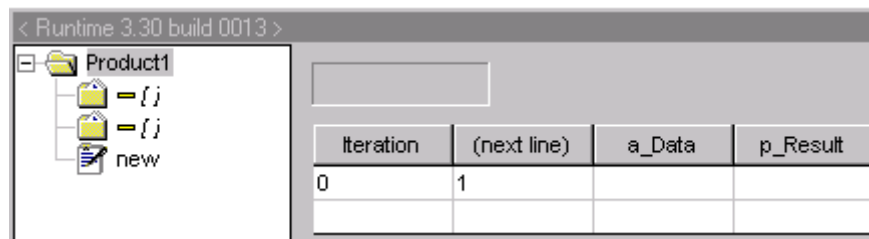
Property for	Column 3
Title	a_Data
Width	70
Usage	Output
<i>Property</i>	p_Data()

Column 4 is the column for displaying result p_Result().

Property for	Column 4
Title	p_Result
Width	70
Usage	Output
<i>Property</i>	p_Result()

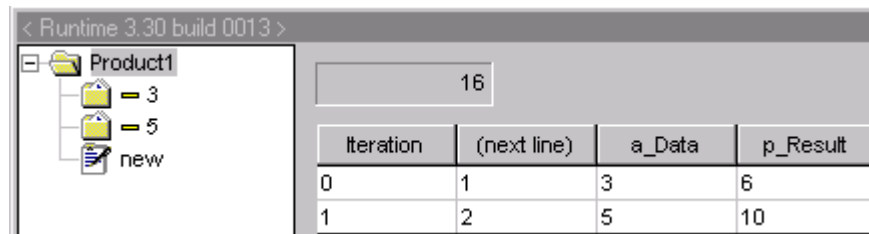
In the runtime layout:

The initial Grid with 2 default iterations.



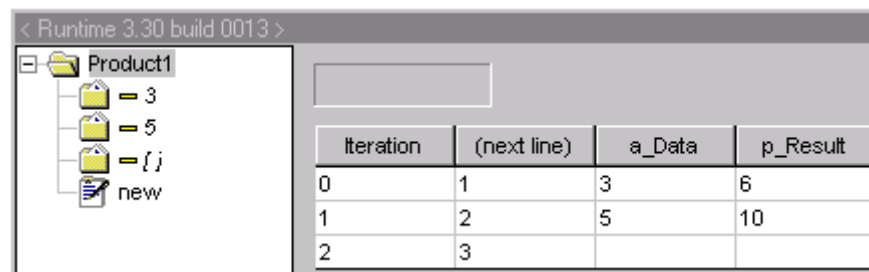
Iteration	(next line)	a_Data	p_Result
0	1		

Grid after data entered.



Iteration	(next line)	a_Data	p_Result
0	1	3	6
1	2	5	10

Elements added or deleted are shown in the Grid:



Iteration	(next line)	a_Data	p_Result
0	1	3	6
1	2	5	10
2	3		

Multiple pages with Extended Grid

Model components.

Layout components.

Adding/deleting array members.

Model components

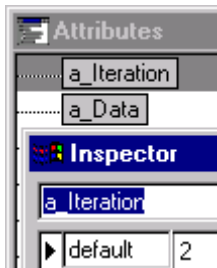
In addition to the requirements for multiple pages without a grid, with an ExtendedGrid the model requires the following:

- Iteration attribute with default.
- Attributes for data input.
- Properties that provide an indexed and non-indexed result.

Example

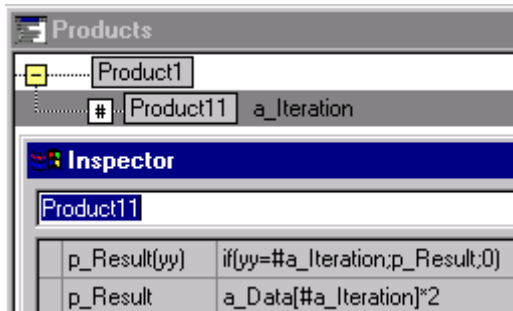
Iteration attribute default 2.

Attribute a_Data is the input data for each iteration.



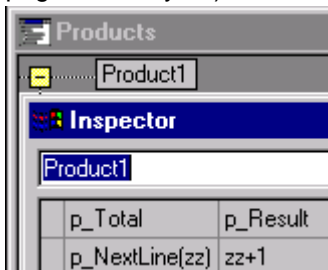
Lower node property p_Result() is required for displaying the result for each iteration in the ExtendedGrid.

(p_Result is not required for the ExtendedGrid, but rather provides the result for each iteration that will be summed to a single result within the model).



Upper node property p_NextLine() is required for generating the iteration index for the next line in the ExtendedGrid.

(p_Total is not required for the ExtendedGrid, but rather is the result for the ResultField in the upper page in the layout).



See the next page for an example layout that uses the above example model.

Layout components

In addition to the requirements for model subnodes, the layout requires the following definitions for the ExtendedGrid:

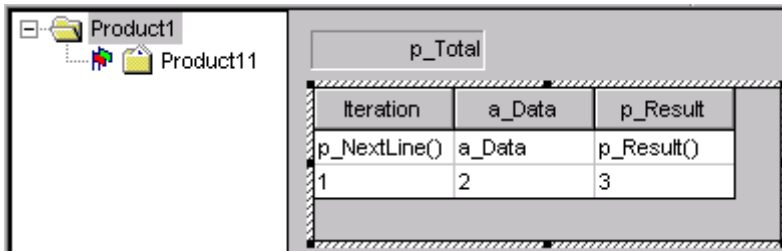
- # of columnn

- Iteration attribute (ExtendedGrid property <Count attribute>)
- Next line column and property
- Data input column and attribute (the data input column is specified as an "input" column).
- Result column and property.

Example

Assume the model described on the previous page.

Assume the following layout with an ExtendedGrid:



<Column> = 3 specifies that the ExtendedGrid contains 3 columns.

a_Iteration is the count attribute.

Type	Extended Grid
Columns moveable	Yes
Column	3
Count attribute	a_Iteration

Column 1 contains the iteration.

Type	Extended Grid
Columns moveable	Yes
Column	3
Count attribute	a_Iteration

Column 2 is the column for data input.

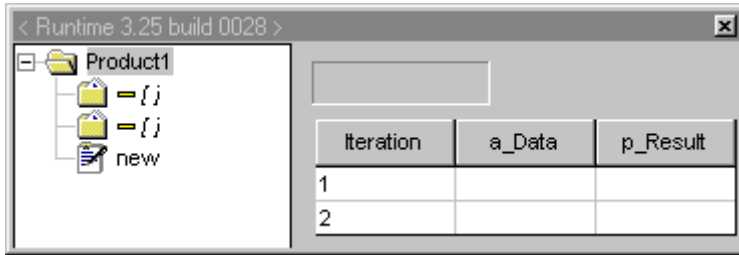
Property for	Column 2
Title	a_Data
Width	70
Usage	Input
Type	Text
Attribute	a_Data
<i>Indexed</i>	Yes
<i>Compute</i>	OnChange
<i>Strict check</i>	No
<i>ChkValidation</i>	Yes
<i>Visible</i>	Yes
<i>ChkOtherVisibility</i>	No
<i>ChkOwnAvailability</i>	None
<i>ChkOtherAvailability</i>	No
<i>ComputeResult</i>	Yes
<i>ChkDynamic</i>	No

Column 3 is the result column.

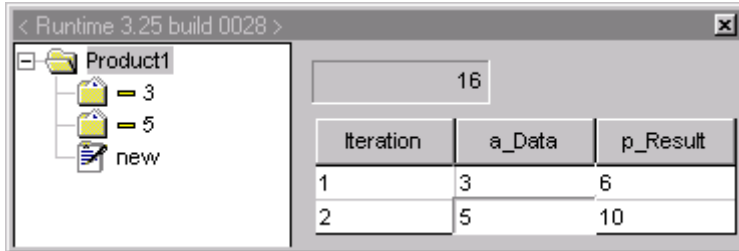
Property for	Column 3
Title	p_Result
Width	70
Usage	Output
Type	Text
Property	p_Result()
<i>Indexed</i>	Yes

In the runtime layout:

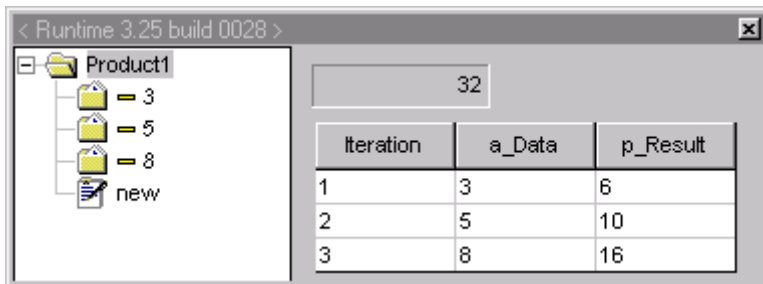
The ExtendedGrid has 2 default rows.



With entered data.



Elements added or deleted using the page frame add/delete functionality are shown in the ExtendedGrid:



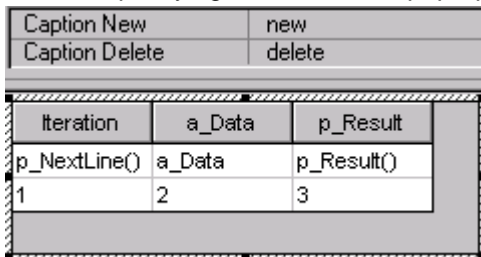
Add/delete array elements

Elements in the array can be added/deleted directly from within the ExtendedGrid by simply right-clicking on the row to delete or on any row (for adding a row) and selecting from the pop-up menu.

Example

Model

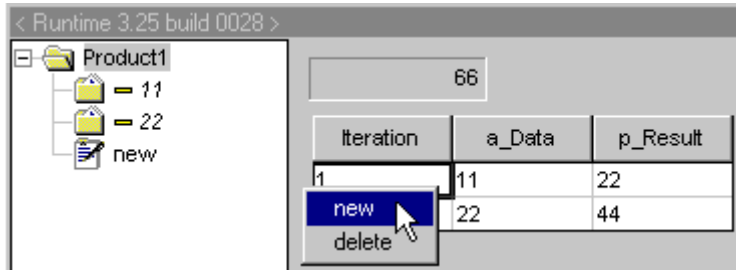
Assume the model described on the previous page with properties <Caption New> and <Caption Delete> specifying the text for the pop-up menu:



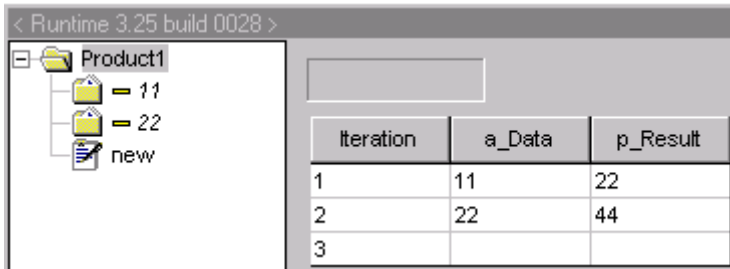
Run-time layout

In the runtime layout:

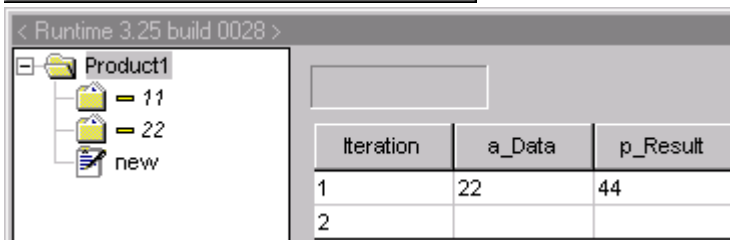
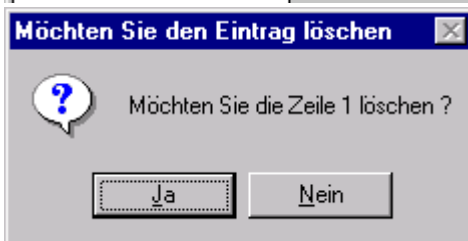
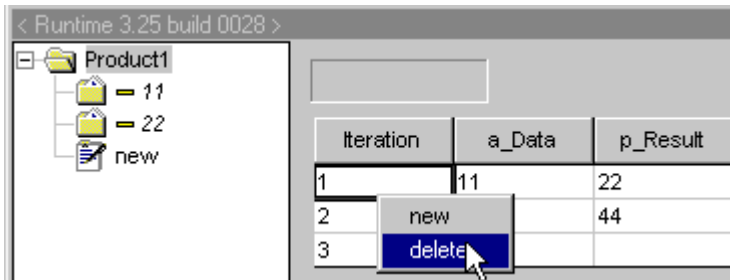
Right-clicking on iteration 1 opens the pop-up menu:



Clicking on "new" adds a new iteration at the end of the ExtendedGrid. Note that the iteration list in the Frame is not updated.



Right-clicking on row 1, clicking on "delete" in the pop-up menu and clicking on "Yes" in the resulting message dialog deletes row 1.



Subdialogs

Secondary Dialogs can be understood visually as an application within an XPL-application. That means, that the same information structure Application/Area/Page/Element can be used to describe and maintain them. Each secondary dialog may be considered like an XPL-application described by its own VPL-file, so we can speak about nested XPL-applications. Maximal nesting level (theoretically unlimited) may be limited on some reasonable number (e.g.16).

VP/MS Designer Runtime will support modal secondary dialogs only. Modal secondary dialogs are "application modal" for the XPL-application. When a modal secondary dialog "pops up", the underlying XPL-application isn't accessible as long as this secondary dialog is open. The intercommunication

with the application model (VP/MS-model) is restricted to the information and elements in the secondary dialog (e.g. result fields of the underlying XPL-application are not computed/updated as long as the secondary dialog is active, they are only computed automatically when the secondary dialog is closed with the OK-Event and if the application property "Calculation" is set to "automatic"). Subdialogs (sub-layouts) can be opened (and closed) from a layout.

- Open
- Close

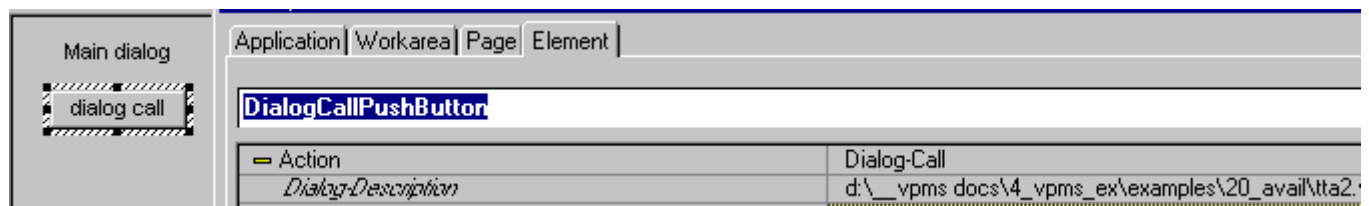
Subdialog: Open

A subdialog is opened by clicking on a PushButton that specifies the .vpc (compiled layout) to open.

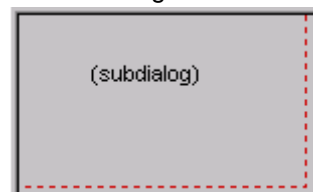
Example

The following layout (tta1.vpl) is the main dialog and contains a PushButton with:

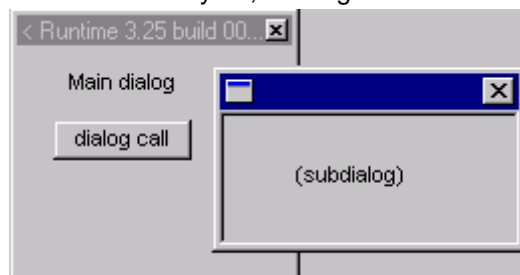
- <Action> = <Dialog-Call>
- <Dialog-Description> = directory and filename of compiled layout for subdialog (<tta2.vpc>)



The following is the subdialog (tta2.vpl):



In the runtime layout, clicking on the Pushbutton would open the subdialog:



Subdialog: Close

A subdialog can be closed by:

- Clicking on an Accept PushButton (the changes made in the subdialog are accepted).
- Clicking on a Cancel PushButton (the changes made in the subdialog are not accepted).
- Clicking on the Windows dialog-close button in the upper right corner of the subdialog.

Subdialog: Accept

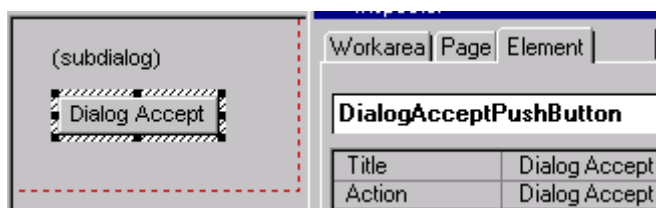
The changes in a subdialog can be accepted by clicking on a PushButton with property <Action> = <Dialog Accept>.

Logical event: Accept. Every data manipulation (in general every user interaction) that happened from the time when the dialog was invoked is accepted. The dialog will be closed and dialog control goes back to the place in the underlying XPL-application from where the dialog was initiated. If the application property calculation; is set to automatic, the XPL-application is computed (intercommunication with the application model). If the application property calculation is set to manual, the XPL-application is not computed only checked.

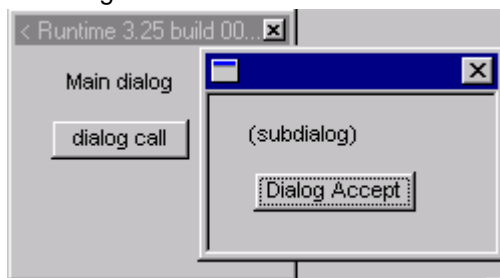
Example

The following layout (tta2.vpl) is the subdialog and contains a PushButton with:

- <Action> = <Dialog Accept>



In the runtime layout, clicking on the Dialog Accept Pushbutton would close the subdialog and accept all changes:



Subdialog: Cancel

The dialog can be closed without accepting changes by clicking on a PushButton with property <Action> = <Dialog Cancel>.

Logical event: Cancel. Every data manipulation that happened from the time when the dialog was invoked is canceled by the Designer Runtime. The dialog will be closed and dialog control goes back to the place in the underlying XPL-application from where the secondary dialog was initiated. The status of the XPL-application is exactly the same as before the secondary dialog was called. The logical events Accept and Cancel are mapped to the standard user interactions for secondary dialogs (keyboard Esc, the close symbol x in the window title and so on) as well as to specific Pushbutton actions (see below).

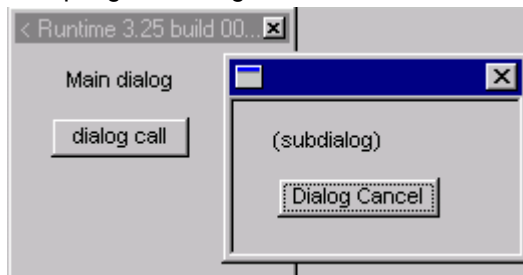
Example

The following layout (tta2.vpl) is the subdialog and contains a PushButton with:

- <Action> = <Dialog Cancel>



In the runtime layout, clicking on the Dialog Cancel Pushbutton would close the subdialog without accepting the changes:



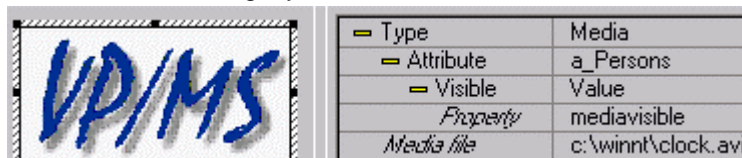
Media element

The following types of multimedia elements can be displayed in the Designer:

- avi
- bmp
- jpg
- tif
- wav

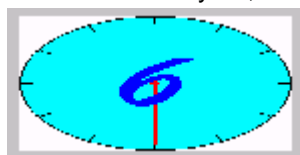
Example

Assume the following layout Media element:



The element is visible if model attribute <a_Persons> property <mediavisible> = 1 (any audio supplied by the element is always played regardless of the visibility).

In the runtime layout, the element would appear stretched to the border defined by the layout element:



The element stops playing if a different page is selected. The element starts playing from the beginning if the page is reselected.

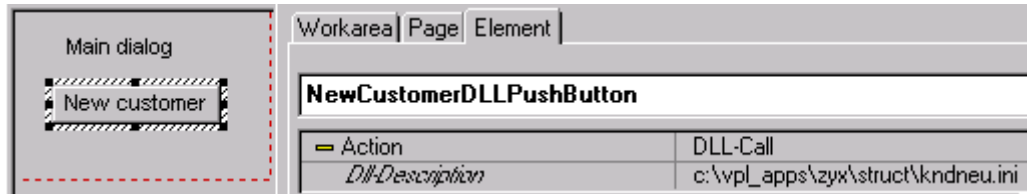
DLL call

A DLL can be called by clicking on a PushButton that specifies the .ini file that specifies the DLL interface.

Example

The following layout contains a PushButton with:

- <Action> = <DLL-Call>
- <Dll-Description> = directory and filename of .ini file that specifies DLL interface (<kndneu.ini>)



Clicking on the PushButton in the runtime-layout will start the DLL.

Layout help / tooltips

Designer supports

- Help
- Tooltips

Layout help

A help topic ID can be specified for each element with the element property <HelpID>.

Example

The EntryField in the following layout has <HelpID> = 123.



Tooltips

2 types of tooltips exist.

- Static tooltips are defined in the layout.
- Dynamic tooltips are defined in the model or in the layout.

Static tooltips

The text for a static tooltip is defined in the layout.

Components with static tooltips

The following components can have ONLY static tooltips:

- ExtendedGridList
- Graphics

- GridList
- Page
- ResultField
- Workarea

The following components can have static OR dynamic tooltips:

- CheckBox
- ComboBox
- EditField
- Label
- Multimedia
- PushButton
- RadioButton
- RadioGroup

Defining a static tooltip

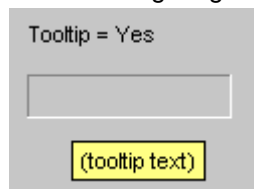
A static tooltip can be defined in 3 different ways.

Method 1 (for components that can have only static tooltips)

- Property <Tooltip> is set to Yes.
- Property <Tooltip text> is set to the tooltip text.

Example:

The following diagram shows a result field with <Tooltip text> = "(tooltip text)".

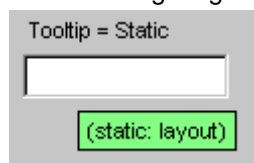


Method 2 (for components that can have dynamic tooltips)

- Property <Tooltip> is set to Static.
- Property <Tooltip text> is set to the tooltip text.

Example:

The following diagram shows an EntryField with <Tooltip text> = "(static: layout)".

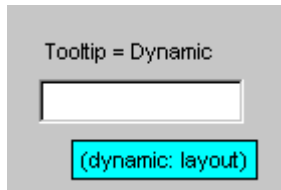


Method 3 (for components that can have dynamic tooltips)

- Property <Tooltip> is set to Dynamic.
- Property <Tooltip text> is set to the tooltip text.
- The source attribute for the component does not have the attribute property <Tooltip> defined (in the model).

Example:

The following diagram shows an EntryField with `<Tooltip text> = "(dynamic: layout)"`. The attribute for the EntryField does not have the property `<Tooltip>` defined in the model.



Dynamic tooltips

The text for a dynamic tooltip is defined in the model.

Components with dynamic tooltips

The following components can have dynamic tooltips:

- CheckBox
- ComboBox
- EditField
- Label
- Multimedia
- PushButton
- RadioButton
- RadioGroup

Defining a dynamic tooltip

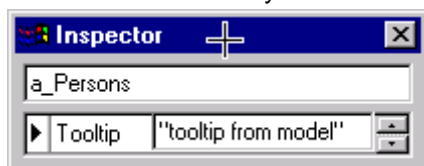
- Property `<Tooltip>` is set to Dynamic.
- In the model: For the component attribute: Attribute property `<Tooltip>` is set to the tooltip text (the value of the property can be dynamic).

Example:

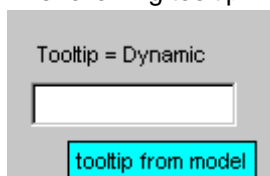
An EntryField has:

- Property `<Tooltip>` = Dynamic.
- Property `<Source>` = `a_Persons`.

In the model for the layout the attribute `a_Persons` has the property `<Tooltip>` = "tooltip from model".



The following tooltip would be displayed in the layout:



Layout comment / print / screen shot

Comments (version information) can be added for all layout components (application, workarea, page, and element).

The contents of the layout can be written to file by printing (CRF preparation) the layout.

Any number of layout pages can be saved to individual bmp files using export screen shot.

Comments

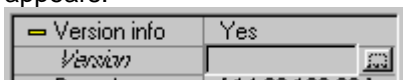
Commentary can be added for any layout component if comments for the component are enabled (default setting for enabling comments is set in the Options dialog) with the version info property.

- Enabling comments (versioning)
- Adding (modifying) comments
- Deleting (disabling) comments

Enabling comments (versioning)

To enable comments for a component:

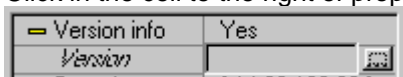
- Select the component.
- Set property Version info for the component to Yes. Note that the subproperty Version appears.



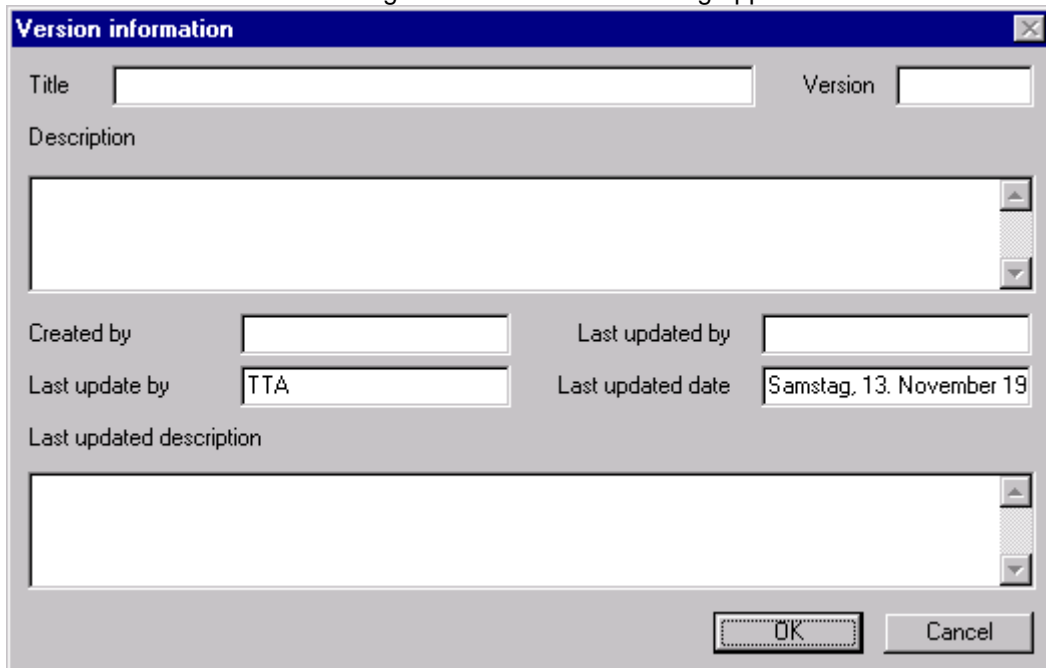
Adding (modifying) comments

To add or modify comments for the component:

- Click in the cell to the right of property Version. Note that a button appears.



- Click on the button. The following Version information dialog appears:



The dialog box is titled "Version information" and contains the following fields:

- Title: [Empty text box]
- Version: [Empty text box]
- Description: [Large empty text area]
- Created by: [Empty text box]
- Last updated by: [Empty text box]
- Last update by: TTA
- Last updated date: Samstag, 13. November 19
- Last updated description: [Large empty text area]

Buttons: OK, Cancel

- Add or modify the information in the Version information dialog.
Note: The fields <Last update by> and <Last updated date> will be filled automatically.
- Click OK to close the dialog. Note that the information entered for "Version" (in this example "(version)") appears in the Inspector.



Disabling comments (versioning)

To disable comments for a component:

- Select the component.
- Set property Version info for the component to No. A message warning that all version information will be lost appears.
- Click Yes. Note that the subproperty Version disappears.

Layout print (CRF Preparation)

A report can be generated that contains a description of the layout components. Report concepts include:

- IFOS settings
- Report format
- Report content
- Report range
- Report sorting
- Report output

The documentation of the user interface (the VPL-definitions) will be done by the new Print-action. The print action can be initiated either by the new menu item File Printing or by the appropriate button in the toolbar.

Executing the print action causes the VPMS Designer implicitly to save the VPL file and then to write all information in the VPL file into formatted ASCII files, that can be read by IFOS (Note: all dialogs will be dumped also each as a single bitmap).

After this information is written to disk into these files, the command file for IFOS will be produced and IFOS will be initiated (new parameters about the configuration of IFOS are added in the INI-file of the VPMS Designer for this purpose).

Depending of report purpose, different kind of element list sorting may be necessary (sorting on elements or attribute names, or pages names). To provide report flexibility IFOS report generator is used for XPL-application reporting. That allows to have different report formats for different kinds of sorting. Every time an XPL-application report generated Designer produces a temporary file containing information about application. This file is an ASCII-file then been used together with an IFOS report template for report producing.

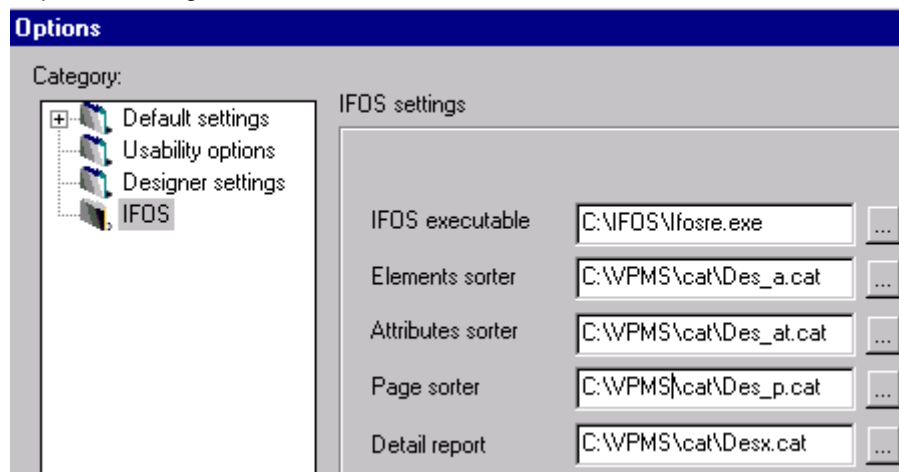
IFOS settings

The following IFOS components are used for generating reports:

- Report editor
- Report viewer
- .cat files

IFOS: Report editor

The report editor (Ifosre.exe) is required for generating the report and must be specified in the "Options" dialog:



IFOS: Report viewer

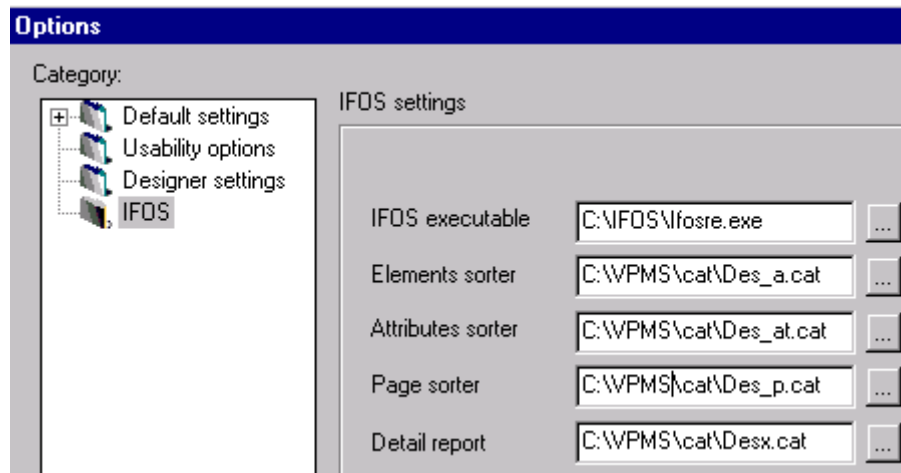
The generated report is automatically opened in the IFOS ReportViewer when the report is output to the screen.

IFOS: .cat files

The following lists the .cat files supplied for generating reports and the function of each file:

- des_p.cat: Table report sorted by page name.
- des_a.cat: Table report sorted by element name or element order..
- des_at.cat: Table report sorted by element attribute <Source> value.
- desx.cat: Text report layout.
- area.cat: Text report workarea.
- page.cat: Text report page.
- element.cat: Text report element.
- colinfo.cat, graphic.cat, tip1.cat ... tip6.cat: Text report properties.

Some of the above files must be specified in the Options dialog:



Report format

The following report formats are available:

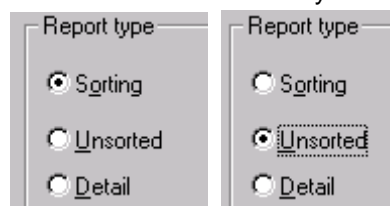
- Table
- Text

Report format: Table

A report in table format contains a 4-column table with the following columns:

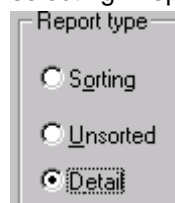
- Workarea name
- Page name
- Element name
- Element property <Source> value

Table format is selected by selecting either "Sorting" or "Unsorted" as the report type.



Report format: Text

A report in text format (no tables) with a line for each property in the selected range is generated by selecting "Report type" radio button "Detail":



Example

The following is a typical page for a generated report:

des.x.cat VP/MS-Designer-Anwendung
C:\VPMS\Designer33\build013\version2\cat\3_15_3.vpl

Anwendungseigenschaften:

Versions-Info No
 Titel:
 Breite: 294
 Höhe: 154
 Format: Individual
 Produktmodell: 3_10_1_1_table.vpm
 Stil: Notebook
 TAB-Font:
 Berechnung: Automatic
 StartAttribut: No
 StartName <Empty>
 Standardschrift: MS Sans Serif, 8
 Farbe aktives Element: (255,255,255)
 Externe DB No
 Kopfzeile Yes
 Fusszeile No

Report content

A report can contain the following information:

- Names of workareas
- Names of pages
- Names of elements
- Element attribute <Source> value
- (text report only) All properties (for application/workareas/pages/elements).
- (text report only) Comments (version information) for all properties (if the checkbox "Version information" is checked).

Examples

A typical table report with the names of workareas / pages / elements / <Source> attributes:

des.p.cat Elementliste der VP/MS-Designer-Anwendung

(Sortiert nach Bereich und Seite)

Bereich	Seite	Element	Attribut
Area1	Area1Page1	A1P1Pushbutton	PagePushbutton
Area1	Area1Page1	AppHeaderLabel	HeaderLabel
Area1	Area1Page1	A1FooterPushbutton	FooterPushbutton
Area2	Area2Page1	A2GroupBox	PageGroupBox

The following is the first page of a text report that describes all properties and includes component commentary (version information):

des x.cat VP/MS-Designer-Anwendung
C:\VPMS\Designer33build013version2\cat3_15_3.vpl

Anwendungseigenschaften:

Versions-Info	No
Titel:	
Breite:	294
Höhe:	154
Format:	Individual
Produktmodell:	3_10_1_1_table.vpm
Stil:	Notebook
TAB-Font:	
Berechnung:	Automatic
StartAttribut:	No
StartName	<Empty>
Standardschrift:	MS Sans Serif, 8
Farbe aktives Element:	(255,255,255)
Externe DB	No
Kopfzeile	Yes
Fusszeile	No

Versionsinformationen der Anwendung:

Titel:
 Versionsnummer:
 Erstellt von:
 Erstellt am:
 Beschreibung:

Letzte Änderung von:
 Letzte Änderung am:
 Beschreibung letzte Änderung:

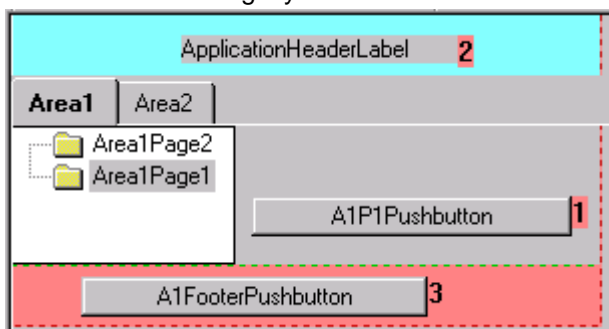
Report range

The report contents can be limited to components from one of the following "ranges":

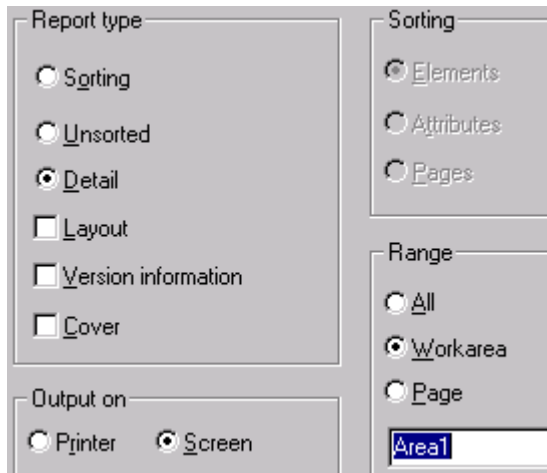
- Application (all components in the layout)
- (text format only) Workarea (all components in the selected workarea)
- (text format only) Page (all components in the selected page)

Examples

Assume the following layout:



Selecting the following print options:



would generated a text format report for:

- The application properties.
- Properties for components in workarea "Area1".

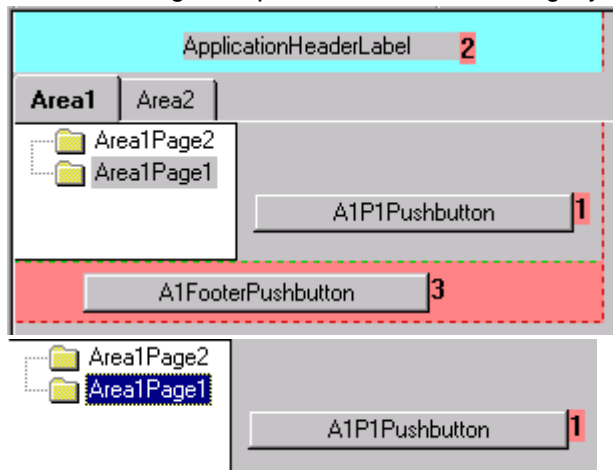
Report sort

The report contents can be sorted according to the following:

- (table format only) Ascending alphabetical order based on 1 of the following:
 - Page names.
 - Element names.
 - Element attribute <Source> value.
- Order of the workareas/pages/elements/properties in the layout/Inspector.

Examples

In the following examples assume the following layout:



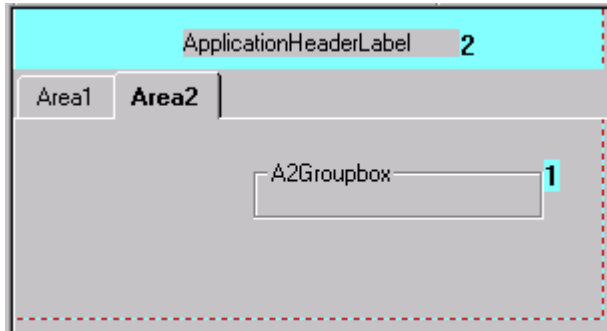


Table format / sort by page names

CRF preparation dialog:



Resultant report:

des_p.cat Elementliste der VP/MS-Designer-Anwendung
(Sortiert nach Bereich und Seite)

Bereich	Seite	Element	Attribut
Area1	Area1Page1	A1P1Pushbutton	PagePushbutton
Area1	Area1Page1	AppHeaderLabel	HeaderLabel
Area1	Area1Page1	A1FooterPushbutton	FooterPushbutton
Area2	Area2Page1	A2Groupbox	PageGroupbox

Table format / sort by element names

CRF preparation dialog:

Resultant report:

Table format / sort by element attribute <Source> value

CRF preparation dialog:

Resultant report:

Table format / sort by element order

CRF preparation dialog:

Resultant report:

Text format / sort by workareas/pages/elements/properties order

CRF preparation dialog:

Resultant report:

Page1:

Page2:

Page3:

Page4:

Report output

The generated report can be output to:

- Screen
- Printer

Report output: Screen

When the report is output to the screen, the report is opened in the Report Editor. There are several options available within the ReportEditor, including the following:

- Saving the report to an .rg file.
- Printing the report.

Click the "Help?" button in the ReportEditor for online help.

Report output: Printer

The report can be sent directly to a printer.

Layout screenshot

Workareas and pages can be selected individually for export to individual .bmp files.
See also:

- Screen shot dialog
- Opening screen shot dialog from menu File / Export screen shot

Example

Assume the following layout:

Selecting from the main menu "File / Export screen shot" opens the following dialog:

Selecting each file and clicking the ">>" button moves the files into the right half of the dialog. Clicking "OK" then displays the following dialog:

The following 2 .bmp files were generated in the same directory as the .vpl file:

- scr_W2Page1.bmp
- scr_Workarea1().bmp

Layout test

The layout can be tested without VFrame using the layout test application (vpmste32.exe).

The test routine can be started by performing any 1 of the following:

- Selecting from the main menu File / Test.
- Clicking on the Test icon.
- Double clicking on the compiled layout file (.vpc) (assuming that designer test is registered as the application for opening files with the .vpc extension).

Common tasks

The following tasks are required often while working with Designer.

- Versioning of layout components and the layout itself.
- Searching for layout components via element or attribute names.
- Aligning the elements in a page.

Task: Versioning

A layout should be versioned at regular intervals.

The versioning process consists of the following:

- Commenting parts.
- Creating a versioned copy of the layout.

Task: Searching

A layout can be searched according to the following:

- Element name
- Attribute name

Task: Aligning

The elements in a layout can be aligned using the alignment tools.

Examples

Each of the following example layouts demonstrates various ways to implement certain functionality in Designer. The layout and the required model for each example are in a subdirectory of directory /docs/english/designer/examples on the CD ROM.

- Tabulator.
- RadioGroup.
- ComboBox.
- Conversion.
- Data indicator.
- Visualize.
- Dynamic.
- Visibility.
- Multiple.
- Page autostart.
- Secondary dialog.
- Compute result.
- Expand/collapse page tree.
- Availability.
- Strict check.
- Grid.
- Wizard.

Example: Dynamic drop-down list (combobox)

Location on CD-ROM

\\docs\\english\\examples\\combodyn (combodyn.vpl, combodyn.pms)

Description

This example demonstrates how to create comboboxes with dynamic content. When the car manufacturer is chosen from the upper combobox, then the available models in the lower combobox are only for that manufacturer.

Example: Radiogroup

Location on CD-ROM

\\docs\\english\\examples\\radiogroup (radiogr.vpl, radiogr.pms)

Description

The Radiogroups serve mainly to display contents of the tables defined by means of VP/MS-Workbench. The displaying can be carried out both in the vertical form and in the horizontal one. If in Designer-Inspector "Number of columns" for any Radiogroup has value of 1 (default value) the vertical form of displaying is used, otherwise the horizontal one is used. In case of the horizontal displaying of the Radiogroup it is necessary to take into account that "Number of columns" corresponds to the number of displayed lines of the table. In case of the vertical displaying it is necessary only to set correct height so that to put all displayed elements inside. The name of the Workbench-attribute with which the displayed table is associated should be specified as "Source" for the Radiogroup in Designer-Inspector. If default value was assigned to the attribute in Workbench this value will be automatically transferred to Designer Runtime.

Example: Combobox

Location on CD-ROM

\\docs\english\examples\combobox (combobox.vpl, combobox.pms)

Description

Comboboxes, as well as the Radiogroups, serve mainly to display contents of the tables defined by means of VP/MS-Workbench. The name of the Workbench-attribute with which the displayed table is associated should be specified as "Source" for the Combobox in Designer-Inspector. If default value was assigned to the attribute in Workbench this value will be automatically transferred to Designer Runtime.

At the startup of any XPL application Comboboxes are filled with the contents of the lists. If the lists are long this process can last for some time, especially if the contents of the list is read out from a database. To reduce starting up time for the application it is possible to specify the Property DYNCOMBO for the Workbench attribute associated to the table. DYNCOMBO is the signal for Designer Runtime to fill the list only when the Combobox is activated for the first time. Thus the time of startup is certainly reduced.

If default value was assigned to the Workbench attribute the related record in Designer Runtime can not be displayed until the list is filled. The attribute's Property LONGTEXT is responsible for that. LONG TEXT returns to Designer Runtime the text, which should be displayed in the Combobox so long as there is no list there.

Java Applets: Here the Property DYNCOMBO is ignored, and Comboboxes are always filled at startup of the application.

Example: Conversion list

Location on CD-ROM

\\docs\english\examples\conversion (conversi.vpl, conversi.pms)

Description

Conversion Lists are usually used to set up certain values in some input fields or to return their default values. The conversion can be performed when pressing a Pushbutton, switching on/off a Checkbox or clicking Radiobuttons. The Property "Conversion List" in Designer Inspector for all mentioned Designer elements serves for these purposes. For Checkboxes the Property "OFF Conversion List" is added to provide an accept to another list of conversion when switching off the Checkbox. The attributes of the conversion are defined in VP/MS Workbench and should be associated with the tables where the actual lists of conversion are. The first column of the table should contain a list of

attribute names, which are transformed by Designer Runtime after definite conversion rules. These conversion rules should be specified in the second column of the table.

It is necessary to take into consideration that the routine use of one Conversion List to set new values and another Conversion List to set the default values for 2 Radiobuttons make sense only if they are incorporated in one group in Designer Inspector through Subproperty "Group".

Example: Data indicator

Location on CD-ROM

\\docs\english\examples\dataindicator (di.vpl, di.pms)

Description

By means of Data Indicators the user can define either the relevant data was entered or no in the certain areas/pages. It is helpful when working with voluminous applications where there are many areas and pages.

In Designer Inspector the Property "DataIndicator" " for the area or the page must be set to "ON" for those areas (Notebooks) and pages (Frames) which should be supplied with Data Indicators. After it has been done, two flags appear in front of the corresponding Notebook heading or Frame heading. They let us know that during the program execution the Data Indicator for the area or the page is used.

In order to use the Data Indicator the Properties of the VP/MS Workbench attributes should have the values which define under what circumstances the Data Indicator is set to ON or to OFF during the running of the program. This Properties should be specified in Designer Inspector as "DI Rule". There should be either written down a complete VP/MS name <Attribute>. <Property>, or only the name of an attribute (in this case the standard Workbench property "DataIndicator" is taken). Properties which used as "DI Rule" should have the value of 0 or of 1. If during the program performance the decision rule (DI Rule) is assigned the value of 1 the corresponding Data Indicator is switched on, if its value is 0 it is switched off.

Through the Subproperties "IN Symbol" and "OUT Symbol" in Designer Inspector it is possible to define what Bitmaps are used when displaying of the Data Indicator during of the program performance (Bitmaps by default: "+" and "-"). The library of Bitmaps for the Data Indicators is together with VP/MS Designer delivered .

Example: Visualize

Location on CD-ROM

\\docs\english\examples\visualize (visualiz.vpl, visualiz.pms)

Description

Some Notebook areas and Frame pages during the program performance can be visualized(displayed) or devisualized(hidden). This opportunity of Displaying / Hiding areas and pages can be defined for Radiobuttons and Checkboxes. It is realized through Subproperties "Visualize" and "Devisualize" in Designer Inspector. After opening the dialog "Visualize" it is possible to define what areas and / or pages will display on the screen when activating appropriate Radiobuttons or Checkboxes. When switching off the Radiobuttons or Checkboxes the visualized area and / or pages are again hidden. The same is for the dialog "Devisualize". But here it is necessary to select the areas and / or the pages which should be removed from the screen when activating appropriate Radiobuttons or Checkboxes.

Attention: When switching on or switching off any Radiobutton or Checkbox not all Frame pages, which are available on one of the Notebook pages, may be removed from the screen. It would cause a collapse of the program.

Example: Dynamic

Location on CD-ROM

\docs\english\examples\dynamic (dynamic.vpl, dynamic.pms)

Description

The Standard-Property "Dynamic" serves for updating attribute / group of attributes, used as input field(s) in VP/MS Designer, when another attribute / group of attributes, to which this attribute / group of attributes was connected by means of "Dynamic", is changed. This functionality is accessible for use only if in the file "win.ini" in the section [VPMS] refresh=0. If refresh=1 then the Standard-Properties "Refresh/Recalc" are processed instead of "Dynamic".

The Property "Dynamic" for the appropriate attributes is defined in a window "Attribute" of VP/MS-Workbench. For each attribute with "Dynamic" the list of attributes should be defined. The attributes in this list are immediately recalculated when the attribute with "Dynamic" is changed. This process is carried out on the basis of the rules, which are defined by means of the Standard-Property "Default" for the attributes updated with the help of the Property "Dynamic". Here all permissible for VP/MS Workbench interrelations can be considered. While using "Dynamic" it is necessary don't allowed giving the rise of an infinite cycle, that can result in the suspending of an application "Dynamic" processing of a GUI element is carried out by means of the Subproperty "CheckDynamic" which can be switched on or switched off.

It is recommended to set the Subproperty "CheckDynamic" to "NO" for the inputs fields which contents don't affect the dynamic status of other GUI elements.

Example: Visible

Location on CD-ROM

\docs\english\examples\visibility (visibili.vpl, visibili.pms)

Description

Visibility of visual elements can be controlled both in VP/MS Workbench and in VP/MS Designer. In VP/MS Workbench it is carried out through the specially reserved Property "Visible", which can be defined for an attribute in a window "Attribute". It determines under what circumstances the attribute becomes visible or invisible.

In VP/MS Designer for the majority of GUI elements there are 2 additional Subproperties "Visible" and "CheckOtherVisibility", which are displayed in Designer Inspector after defining of the source of the corresponding element.

Subproperty "Visible" replaces the old Subproperty "CheckOwnVisibility" and can assume values "Yes", "No", "Check" or "Value". If the value "No" is assumed, the corresponding GUI element during the program performance is always invisible. If "Visible" = "Yes" the GUI element is always visible. In these two cases no visibility checking are carried out during the program performance. In other two cases the visibility of a GUI element is checked.

When "Visible" = "Check" (for the result fields such opportunity is absent) visibility of the corresponding GUI element is checked through the attribute, for which a Standard Property "Visible" was defined in

VP/MS Workbench. And depending on the rule defined by means of the Property "Visible" a GUI element during program performance becomes visible or invisible. It takes place, certainly, only when the other GUI elements (their names are specified in a Visible condition in WorkBench), on which the visibility of the given GUI element depends have the Subproperty "CheckOtherVisibility" set to "Yes".

If the Property "Visible" set to "Value", it is possible to indicate as value, a name of an attribute, or a Property or a function defined in the product model. If during the program performance the attribute, the Property or the function assumes value TRUE (1), the corresponding GUI element becomes visible. Otherwise the GUI element is removed from the screen. It is necessary to take into account that only the functions without call parameters can be used for visibility check. Besides, the visibility check while using the attribute as the value is carried out only when the other GUI element on which attribute the visibility of this GUI element depends, have the Subproperty "CheckOtherVisibility" set to "Yes".

For the Subproperty "CheckOtherVisibility" the values "Yes" or "No" can be selected. If set to "No" no visibility checking for other GUI elements which visibility depends on this GUI element is processed at runtime.

If the Subproperty "CheckOtherVisibility" assumes value "Yes", the visibility of other GUI elements is checked but in fact of those only with which Subproperty "Visible" was not assigned "Yes" or "No". Taking into account the productivity reasons it means, that this Subproperty should not assume value "No" for all the GUI elements which do not influence visibility of other GUI elements.

Example: Multiple

Location on CD-ROM

\\docs\english\examples\multiple (multiple.vpl, multiple.pms)

Description

The multiple instances are used when the number of persons or objects should be insured. The logic of repetitive processings (accounts) is described in detail in the Handbook for VP/MS Workbench. Here are considered only special features of VP/MS Designer.

To represent repeated Workbench-instances in VP/MS Designer are used the Frame pages the Property "Dynamic" for which has value "Yes". The Subproperty "Identification" indicates which Workbench attribute is selected as the identifier for the Frame page and is used as heading of the corresponding copy of the page. In the Subproperty "Count attribute" from VP/MS Workbench the corresponding run counter should be copied (this counter is recommended not to be used as a GUI element in VP/MS Designer). The Sub properties "Caption New" and "Caption Delete" mean in each case Frame heading when placing a multiple page copy and the message which will be displayed in the dialogue when removing a multiple page copy.

It is necessary to take into account that the Subproperty "Multiple" should be set to "Yes" for the elements which attributes differ for the insured person or the object. If the Subproperty "Multiple" set to "No" for some elements in Designer Inspector then the same value of the attribute will be used for this elements on all multiple page copies.

Example: Page autostart

Location on CD-ROM

\\docs\english\examples\page_autostart (pageauto.vpl, pageauto.pms)

Description

The situation when the result field and the input fields which are needed for calculations placed on the different Frame pages or Notebook areas happens enough often.

In this case, if the input field, which used for calculating the result, is placed on one page and the result field is placed on another, the corresponding page where the input field is placed is automatically started up. Certainly, it doesn't occur, if this input field has a default value. Then this value is used for calculating the result.

If the input field on the started page is not filled or the contents of the given field is deleted and the user is returned back to the previous page (that is possible in Designer Runtime only at the second attempt), an automatic transition is not possible any more (thus, the infinite cycles are excluded).

It is necessary to pay attention to the value of the Subproperty "ComputeResult" of an element. If this Sub property set to "No" for any of the input fields, no calculating of the result field will be executed after changing of the data in these elements (other pages will not be started up also). This means that for productivity of the work with application this Subproperty should be set to "No" for all input fields having an informative meaning.

Example: Secondary dialogs

Location on CD-ROM

\\docs\english\examples\secondary_dialogs (parent.vpl, child.vpl)

Description

During the running of the program there is an opportunity to get access to another XPL-applications. For this purpose so-called secondary dialogs, which can be activated from a parental application, are used. Each secondary dialog may be consider as the XPL-application and is described by its own VPL-file (like every XPL-application). VP/MS-Designer-Runtime sets no limits to the maximal depth of inclusions.

Designer-Runtime supports only modal secondary dialogs. It means that if the secondary dialog is activated, the parental application becomes inaccessible until the nested dialog will be closed.

The only way to activate the secondary dialog is with a pushbutton. For this purpose the actual pushbut-tons Property "Action" with the value "Dialog-Call" is used.

The Subproperty "Dialog-Description" is necessary to specify the name of the VPL-file, which is considered as the child application. The heading of the child application appears in the headline of the secondary dialog when it called during the program performance. By means of the Subproperty "New Session" is defined whether the secondary dialog uses the same VP/MS-session as its parental application or it opens a new one (it is necessary if the child application uses another VPM-file as a product model). In the Sub-properties "Left" and "Top" a distance from the left top corner of secondary dialog to the left top corner of the parental application is indicated.

The child application can be used, for example, for data transfer to the parental application for result calculating. Therefore 2 Pushbuttons should be defined in the child application. One of the Pushbuttons should serve for accepting the data entered in the secondary dialogue. It is achieved if the value "Dialog Accept" is set in the Pushbutton-Property "Action" in Designer-Inspector. By pressing the second Pushbutton the data entered in the secondary dialog are rejected and not used in the parental application for calculating the result. It is achieved by means setting the Pushbutton-Property "Action" to "Dialog Cancel" .

It is necessary to take into account that the result fields in the parental application are computed automati-cally only when the secondary dialog was closed with result OK and the Application Property "Calculation" for the parental application is set to "Automatic".

Example: Compute results

Location on CD-ROM

\docs\english\examples\compute_results (compute.vpl)

Description

The result fields of the XPL-application can be computed either automatically or manually. It is controlled by the Application Property "Calculation". Using manual computing the values in the result fields are computed only when the corresponding Pushbutton is pressed. To have an opportunity to carry out manual calculation, it is necessary to define a Pushbutton and to set "Compute" to the Element Property "Action". Besides, the Application Property "Calculation" should be set to "Manual". It is necessary to take into account that the Property "Calculation" influences all result fields, i.e. all result fields are computed either automatically or by pressing the Pushbutton. For some input fields it is possible to define, however, through the Element Subproperty "Compute" whether the corresponding fields of result will be computed "OnChange" of an input field or only "OnExit" of input field.

Java Applets: Here computing of the result fields is carried out only after quitting the corresponding input fields regardless of the Element Subproperty "Compute" set to "OnChange" or "OnExit".

Example: Expand/collapse page tree

Location on CD-ROM

\docs\english\examples\expand_collapse_tree (expand.vpl, expand.pms)

Description

If any Frame-page has subpages it is possible to expand or collapse the Tree-View by means of double click by the left button of the mouse.

It can be defined in Designer-Inspector whether at loading of the XPL-application the corresponding Frame-page are automatically expanded and subpages displayed in Tree-View, or this page are collapsed and its subpages in Tree-View remain hidden.

For this purpose the Page Property "Defolding" is used. If the value "Manual" is set for this Property, Tree-View for the page will not be expanded automatically during loading and it can be then expanded only using the mouse. For automatic expanding of the Frame-page it is necessary to set the value "Automatic" (default value) for the Property "Defolding".

Example: Available

Location on CD-ROM

\docs\english\examples\availability (availabi.vpl, availabi.pms)

Description

Availability of the visual elements, unlike the visibility, can be checked only in VP/MS-Designer. For the majority of GUI-elements in VP/MS-Designer are additionally present 2 Subproperties "CheckOwnAvailability" and "CheckOtherAvailability" which are displayed after definition of a source for the corresponding element in Designer-Inspector.

The Subproperty "CheckOwnAvailability" can be set to "No", "All", "List" or "Page". If the Subproperty set to "No" the Availability check of the corresponding GUI-element will not be carried out during the program performance.

If "CheckOwnAvailability" = "All", the GUI-element Availability is checked during the program performance using all result fields that are present in the XPL-application. It means that if for processing of at least one result field in the application a relevant attribute is required by the GUI-element this GUI-element becomes accessible. However, owing to the productivity reasons this value is not recommended to set. When saving Layout with such settings in a references window a reference to elements with "CheckOwnAvailability" = "All" is provided.

If "CheckOwnAvailability" = "List" the Availability of the GUI-element is checked during the program performance only through result fields specified in the Subproperty "List (...; ...)". It means that only these GUI-elements become available, which attributes for processing at least one result field are needed. Owing to the productivity reasons it is recommended to place in the list only one result field.

If "CheckOwnAvailability" = "Page" the Availability of the GUI-element on the page is checked during the program performance through the result field specified in AVAILABLE-Property for the page. It means that if for processing a result field the attribute of the GUI-element located on this page is needed this GUI-element becomes accessible.

It also should be taken into account that in the product model can be additionally the attributes which used for calculating the result. It means that when these attributes assume the certain values the different lists of the attributes can be used for calculating the Property (in VP/MS-Designer as the result fields are used Workbench-Properties). Therefore in Designer is a Subproperty "CheckOtherAvailability", which can take values "Yes" or "No".

If "CheckOtherAvailability" = "No", the Availability check for other GUI-elements during the program performance will not be carried out if the contents of the element changed. Otherwise the availability of the GUI-elements is checked. However if "CheckOtherAvailability" = "Yes" those GUI-elements are checked only, for which the Subproperty "CheckOwnAvailability" set to "Yes".

Owing to the productivity reasons it is recommended to set the Subproperty "CheckOtherAvailability" to "No" for those GUI-elements, whose values do not influence the availability of other GUI-elements.

Example: Strict check

Location on CD-ROM

\\docs\english\examples\strict check (strictch.vpl, strictch.pms)

Description

The Subproperty of an element "Strict check" enables to forbid the leaving of an input field at some conditions. This Subproperty can be set to "Yes" or "No".

If the Subproperty "Strict check" is set to "Yes" and Minimum-check or the Check-check for the GUI-element returns a negative result, this input field can't be left until the contents of the field satisfied the respective conditions. At Minimum-check it is tested whether the entered value takes less positions than was specified in the element Subproperty "Minimum" or no. At Check-check it is checked whether the entered value meets the requirements specified in the Workbench-Property "check" or no. Here it is necessary to take into account that in addition the element Subproperty "CheckValidation" should be set to "Yes" to perform check specified for attribute in the Workbench-Property "check".

If the Subproperty "Strict check" set to "No" the field can be left always, independently of the results of the Minimum- and the Check-checks.

Example: Grid

Location on CD-ROM

\docs\english\examples\grid (grid.vpl, grid.pms, grid1.vpl, grid1.pms)

Description

Grids are used to display data in a tabular form. It can be necessary, for example, to display the data of each instance of a multiple page in a separate row of the table, or to display in each row different results of functions depending on the one variable factor.

The element Properties "Column" and "Key column" in Designer-Inspector define the number of columns and key column number for the grid. By means of the Property "Iteration type" the number of lines in the grid is defined. This number can be either constant (if the Property "Iteration type" is set to "Counter" and in the Subproperty "Iteration count" is defined a constant number) or variable depending on a value of the attribute or the Property (if the Property "Iteration type" is set to "Attribute" or "Property" and in the Sub-property "Attribute" or "Iteration property" is indicated the Workbench-attribute or the Workbench-Property).

In the Property "Property for" is indicated the column, for which the following Properties are defined. In the Properties "Title" and "Width" the heading of the corresponding column of the grid and the width of this column can be defined. In the Property "Usage" for key column the value "Input" and for others columns the value "Output" should be selected.

In the first line of the key column during the program performance the initial value of the attribute defined in Workbench is displayed. This value serves for identification of the grid columns and should be changeable by means of the function defined in Workbench. This attribute is defined in Designer-Inspector in the Column Subproperty "Attribute". By means of the Column Subproperty "Next Line" in Designer-Inspector the function is defined through which key attribute can be transformed. There the grid column number (output column) is indicated, in which the result of the function will be displayed. Usually during the program performance in Designer-Inspector this column is not shown. For this purpose zero width is defined for this column.

For output columns in the Subproperty "Property" the name of the function or the Workbench-Property of the multiple page with the parentheses should be specified. This function / Property should depend only on the one parameter so that it could be identified by means of the corresponding value of the key column. This value of the function / Property is displayed in the line identified in an appropriate way.

Example: Wizard

Location on CD-ROM

\docs\english\examples\wizard (wizard.vpl, wizard.pms)

Description

VP/MS Designer-Runtime does not give a direct opportunity for development of the Wizard-Applications, as, for example, by means of the visualizing / hiding pages by pressing Pushbuttons. Certainly, it is possible to avoid this restriction of Designer-Runtime. For this purpose it is proposed to use Checkboxes for the visualizing / hiding pages, and also to switch on or switch off these Checkboxes by means of using convert attributes assigned to Pushbuttons.

At first in VP/MS Designer should be created the Frame-pages the transition to which will be carried out using Pushbuttons. Then the WorkArea Subproperty "Frame width" for these pages in Designer-Inspector should be set to 0 in order to the Tree-View remains hidden from the user. For each page it is necessary to define the Checkbox and indicate in the Subproperty "Visualize" the page which will appear on the screen when this checkbox is switched on. For all Checkboxes the Subproperty "Visible" has to be set to "No" in order to the user can't see they during the program performance. For each Checkbox the Workbench-attribute, which then is changed by means of the conversion lists during the program performance, should be assumed. The value of 1 should be set by default to the Workbench-attribute for this Checkbox, which makes page visible at the startup of the application. On

each page it is necessary to place Pushbuttons, by pressing on which it would be possible to come over to the certain pages. It is necessary to assign the con-version list to every Pushbutton. The rules in each conversion table in WorkBench should be defined so that by pressing an appropriate Pushbutton the Checkbox is activated making visible the page on which you come over while other Checkboxes remain inactive.

If you perform all actions described above you'll receive a pure Wizard-application from the user point of view .

The note: Using of the conversion attributes was described in detail in the chapter "The Purpose and Using Conversion Lists".

Properties

The properties are accessible via the Inspector.

- Properties listed by component.
- Alphabetical list of all properties.

Properties: Component list

Properties for:

- Application
- Workarea
- Page

Properties for elements:

- EditField
- ComboBox
- CheckBox
- RadioButton
- PushButton
- Label
- Border
- Grid
- ExtendedGrid
- 3D Border
- Line
- Multimedia element
- ResultField
- RadioGroup
- Graphic element

Properties: Application

The following properties can be set for the application:

- Version info
- Version info = Yes . Version
- Title
- Width
- Height
- Format
- Productmodel
- Style
- Style = Notebook . Tab font
- Calculation
- StartAttr
- StartAttr = Constant . Name
- StartAttr = Constant . Value

- StartAttr = Dynamic . Name
- Default font
- AppActiveColor
- Header
- Header = Yes . Height
- Header = Yes . Background color(R,G,B)
- Footer
- Footer = Yes . Height
- Footer = Yes . Background color(R,G,B)

Properties: Workarea

The following properties can be set for the workarea:

- Version info
- Version info = Yes . Version
- Title
- Tab color
- DataIndicator
- DataIndicator = On . DI Rule
- DataIndicator = On . ON Symbol
- DataIndicator = On . OFF Symbol
- HelpID
- Tooltip
- Tooltip = Yes . Tooltip text
- Tooltip = Yes . Background
- Tooltip = Yes . Font
- Style
- Style = Frame . Frame width
- Style = Frame . Frame font
- Style = Frame . FrameMoveable
- Style = Frame . FrameMoveable = Yes . MinFrameWidth
- Style = Frame . FrameMoveable = Yes . MaxFrameWidth
- Header
- Header = Yes . Height
- Header = Yes . Position
- Header = Yes . Background color(R,G,B)
- Footer
- Footer = Yes . Height
- Footer = Yes . Position
- Footer = Yes . Background color(R,G,B)
- Visible
- Visible = Property . Name

Properties: Page

The following properties can be set for a page:

- Version info
- Version info = Yes . Version
- Title
- Tooltip
- Tooltip = Yes . Tooltip text
- Tooltip = Yes . Background
- Tooltip = Yes . font

- HelpID
- Multiple
- Multiple = Yes . Identification
- Multiple = Yes . Count attribute
- Multiple = Yes . Caption New
- Multiple = Yes . Caption Delete
- Defolding
- Order
- Background color(R,G,B)
- AVAILABLE-Property
- DataIndicator
- DataIndicator = On . DI Rule
- DataIndicator = On . ON Symbol
- DataIndicator = On . OFF Symbol
- Data Elements
- Visible
- Visible = Property . Name

Properties: EditField

The following properties can be set for an EditField:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = Text / Currency / Number . Minimum
- Type = Text / Currency / Number . Maximum
- Type = Text . Multiline
- Type = Text / Date / Currency / Number / Mask . Alignment
- Type = Text / Date / Currency / Number / Mask . Strict check
- Type = Date . Format
- Type = Currency . Currency symbol
- Type = Currency / Number . Decimal position
- Type = Mask . Mask
- Source
- Source = [source] . Compute
- Source = [source] . Strict check
- Source = [source] . ChkValidation
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOtherVisibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability = List . List [...;...]
- Source = [source] . Visible = Yes / Check / Value . ChkOtherAvailability
- Source = [source] . Visible = Yes / Check / Value . ComputeResult
- Source = [source] . ChkDynamic
- Default font

- Background color(R,G,B)
- ActivColor
- ActivColor = UserDef . ElemActiveColor
- Element3D
- HelpID
- Level

Properties: ComboBox

The following properties can be set for a ComboBox:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Source
- Source = [source] . Compute
- Source = [source] . Strict check
- Source = [source] . ChkValidation
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOtherVisibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailibility = List . List [...;...]
- Source = [source] . Visible = Yes / Check / Value . ChkOtherAvailibility
- Source = [source] . Visible = Yes / Check / Value . ComputeResult
- Source = [source] . ChkDynamic
- Default font
- Background color(R,G,B)
- ActivColor
- ActivColor = UserDef . ElemActiveColor
- Element3D
- HelpID
- Level

Properties: CheckBox

The following properties can be set for a CheckBox:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip

- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = CheckBox . Title
- Type = CheckBox . Alignment
- Type = CheckBox . Multiline
- Type = CheckBox . Visualize
- Type = CheckBox . Devisualize
- Type = CheckBox . Conversion List
- Type = CheckBox . OFF conversion List
- Source
- Source = [source] . Compute
- Source = [source] . Strict check
- Source = [source] . ChkValidation
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOtherVisibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability = List . List [...;...]
- Source = [source] . Visible = Yes / Check / Value . ChkOtherAvailability
- Source = [source] . Visible = Yes / Check / Value . ComputeResult
- Source = [source] . ChkDynamic
- Default font
- Background color(R,G,B)
- Element3D
- HelpID
- Level

Properties: RadioButton

The following properties can be set for a RadioButton:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = RadioButton . Title
- Type = RadioButton . Group
- Type = RadioButton . Alignment
- Type = RadioButton . Multiline
- Type = RadioButton . Visualize
- Type = RadioButton . Devisualize
- Type = RadioButton . Conversion List
- Source
- Source = [source] . Compute
- Source = [source] . Strict check
- Source = [source] . ChkValidation

- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOtherVisibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability = List . List [...;...]
- Source = [source] . Visible = Yes / Check / Value . ChkOtherAvailability
- Source = [source] . Visible = Yes / Check / Value . ComputeResult
- Source = [source] . ChkDynamic
- Default font
- Background color(R,G,B)
- Element3D
- HelpID
- Level

Properties: PushButton

The following properties can be set for a PushButton:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Title
- Action
- Action = DLL-Call . Dll-Description
- Action = Dialog-Call . Dialog-Description
- Action = Dialog-Call . New Session
- Action = Dialog-Call . Left
- Action = Dialog-Call . Top
- ConvertAttr
- Source
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability
- Source = [source] . ChkOwnAvailability = List . List [...;...]
- Default font
- Element3D
- HelpID
- Level

Properties: Label

The following properties can be set for a Label:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left

- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = Label . Title
- Type = Label . Default font
- Type = Label . Attribute
- Type = Label . Attribute = [attribute] . Visible
- Type = Label . Attribute = [attribute] . Visible = Value . Property
- Type = Label . Attribute = [attribute] . Visible = Yes / Check / Value . ChkOwnAvailability
- Type = Label . Attribute = [attribute] . ChkOwnAvailability = List . List [...;...]
- Type = Label . Alignment
- Type = Label . Multiline
- Type = Label . Background color (R,G,B)
- Level

Properties: Border

The following properties can be set for a Border:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Type
- Type = Border . Title
- Type = Border . Default font
- Type = Border . Attribute
- Type = Border . Attribute = [attribute] . Visible
- Type = Border . Attribute = [attribute] . Visible = Value . Property
- Type = Border . Attribute = [attribute] . ChkOwnAvailability
- Type = Border . Attribute = [attribute] . ChkOwnAvailability = List . List [...;...]
- Type = Border . Background color (R,G,B)
- Level

Properties: Grid

The following properties can be set for a Grid:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Yes . Tooltip text

- Tooltip = Yes . Background
- Tooltip = Yes . font
- Type (cannot be changed)
- Column
- Columns moveable
- Key column
- Iteration type
- Iteration type = Attribute . Attribute
- Iteration type = Property . Iteration property
- Iteration type = Counter . Iteration count
- Visible
- Visible = Value . Property
- Level
- Property for
- Property for = Column [column#] . Title
- Property for = Column [column#] . Width
- Property for = Column [column#] . Usage
- Property for = Column [column#] . Usage = Input . Attribute
- Property for = Column [column#] . Usage = Input . Next line
- Property for = Column [column#] . Usage = Output . Property
- Property for = Column [column#] . Type
- Property for = Column [column#] . Type = Text / Number / Currency . Alignment
- Property for = Column 1 . Type = Number / Currency . Decimalposition
- Property for = Column 1 . Type = Number / Currency . Alignment
- Property for = Column 1 . Type = Currency . CurrencySymbol
- Property for = Column 1 . Font
- Property for = Column 1 . Background color(R,G,B)

Properties: ExtendedGrid

The following properties can be set for an ExtendedGrid:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Yes . Tooltip text
- Tooltip = Yes . Background
- Tooltip = Yes . font
- Type (cannot be changed)
- Columns moveable
- Column
- Count attribute
- Caption New
- Caption Delete
- Visible
- Visible = Value . Property
- Level
- Header font
- Property for
- Property for = Column [column#] . Title
- Property for = Column [column#] . Width

- Property for = Column [column#] . Usage
- Property for = Column [column#] . Usage = Input . Type
- Property for = Column [column#] . Usage = Input . Type = Text / Number / Currency . Minimum
- Property for = Column [column#] . Usage = Input . Type = Text / Number / Currency . Maximum
- Property for = Column [column#] . Usage = Input . Type = Text / Number / Currency / Date / Mask . Alignment
- Property for = Column [column#] . Usage = Input . Type = Text / Number / Currency / Date / Mask . Strict check
- Property for = Column [column#] . Usage = Input . Type = Number / Currency . Decimal position
- Property for = Column [column#] . Type = Number / Currency . Alignment
- Property for = Column [column#] . Type = Currency . CurrencySymbol
- Property for = Column [column#] . Usage = Input . Type = Date . Format
- Property for = Column [column#] . Usage = Input . Type = Mask . Mask
- Property for = Column [column#] . Font
- Property for = Column [column#] . Background color(R,G,B)

Properties: 3D Border

The following properties can be set for a 3D Border:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Type
- Type = 3D-Area . Background color (R,G,B)
- Level

Properties: Line

The following properties can be set for a Line:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Type
- Type = Line . Background color (R,G,B)
- Level

Properties: Multimedia element

The following properties can be set for a Multimedia element:

- Version info

- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = Media . Attribute
- Type = Media . Attribute = [attribute] . Visible
- Type = Media . Attribute = [attribute] . Visible = Value . Property
- Type = Media . Media file
- Level

Properties: ResultField

The following properties can be set for a ResultField:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Yes . Background
- Tooltip = Yes . font
- Tooltip = Yes . Tooltip text
- Type
- Type = Result . Multiline
- Type = Result / Number result / Currency result / Date result / Mask result . Alignment
- Type = Date result . Format
- Type = Currency result . Currency symbol
- Type = Currency result / Number result . Decimal position
- Type = Mask result . Mask
- Source
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Default font
- Background color(R,G,B)
- Element3D
- HelpID
- Level

Properties: RadioGroup

The following properties can be set for a RadioGroup:

The following properties can be set for a RadioGroup:

- Version info
- Version info = Yes . Version

- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Static / Dynamic . Tooltip text
- Tooltip = Static / Dynamic . Background
- Tooltip = Static / Dynamic . font
- Type
- Type = RadioButton . Alignment
- Type = RadioButton . Number of columns
- Source
- Source = [source] . Compute
- Source = [source] . Strict check
- Source = [source] . ChkValidation
- Source = [source] . Visible
- Source = [source] . Visible = Value . Property
- Source = [source] . Visible = Yes / Check / Value . ChkOtherVisibility
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability
- Source = [source] . Visible = Yes / Check / Value . ChkOwnAvailability = List . List [...;...]
- Source = [source] . Visible = Yes / Check / Value . ChkOtherAvailability
- Source = [source] . Visible = Yes / Check / Value . ComputeResult
- Source = [source] . ChkDynamic
- Default font
- Background color(R,G,B)
- Element3D
- HelpID
- Level

Properties: Graphics element

The following properties can be set for a Graphics element:

The following properties can be set for a Graphics element:

- Version info
- Version info = Yes . Version
- Bounds
- Bounds = [top,left,width,height] . Left
- Bounds = [top,left,width,height] . Top
- Bounds = [top,left,width,height] . Width
- Bounds = [top,left,width,height] . Height
- Tooltip
- Tooltip = Yes . Tooltip text
- Tooltip = Yes . Background
- Tooltip = Yes . font
- Type (cannot be changed)
- Identification Grid
- font
- Background color(R,G,B)
- Element3D
- HelpID
- Graphics type
- Graphics type = Line / Vertical bar / Horizontal bar / Vertical 3D bar / Horizontal 3D bar / Vertical Gantt / Horizontal Gantt . X-axis name

- Graphics type = Line / Vertical bar / Horizontal bar / Vertical 3D bar / Horizontal 3D bar / Vertical Gantt / Horizontal Gantt . Y-axis name
- Key column
- Series
- Legend placement
- Level
- Property for
- Property for = Serie [serie#] . Data column
- Property for = Serie [serie#] . Color

Properties: All

The following is an alphabetical list of reserved Designer properties.
The properties are accessible via the Inspector.

- Action
- ActivColor
- Alignment
- AppActiveColor
- Attribute
- AVAILABLE-Property
- Background
- Background color
- Bounds
- Calculation
- Caption Delete
- Caption New
- ChkDynamic
- ChkOtherAvailibility
- ChkOtherVisibility
- ChkOwnAvailibility
- ChkValidation
- Color
- Column
- Columns moveable
- Compute
- ComputeResult
- Conversion List
- ConvertAttr
- Count attribute
- Currency symbol
- Data column
- Data elements
- DataIndicator
- Decimal position
- Default font
- Defolding
- Devisualize
- DI Rule
- Dialog-Description
- DII-Description
- ElemActiveColor
- Element3D
- Font
- Footer
- Format (application)

- Format (element)
- Frame font
- Frame width
- FrameMoveable
- Graphics type
- Group
- Header
- Header font
- Height (application)
- Height (header/footer)
- HelpID
- Identification
- Identification Grid
- Indexed
- Iteration count
- Iteration property
- Iteration type
- Key column
- Left
- Legend placement
- Level
- List (...;...)
- Mask
- MaxFrameWidth
- Maximum
- Media file
- MinFrameWidth
- Minimum
- Multiline
- Multiple
- Name (application)
- Name (workarea/page)
- New Session
- Next line
- Number of columns
- Off conversion list
- Off symbol
- On symbol
- Order
- Position (footer)
- Position (header)
- ProductModel
- Property
- Property for
- Series
- Source
- StartAttr
- Strict check
- Style (application)
- Style (workarea)
- Tab color
- Tab font
- Title
- Tooltip (static/dynamic)
- Tooltip (static)
- Tooltip text
- Top

- Type (entry elements)
- Type (ExtendedGrid column)
- Type (ExtendedGrid)
- Type (GraphicsElement)
- Type (Grid column)
- Type (Grid)
- Type (result elements)
- Type (visual elements)
- Usage
- Value
- Version
- Version info
- Visible (element)
- Visible (workarea/page)
- Visualize
- Width (application)
- Width (Grid/ExtendedGrid)
- X-axis name
- Y-axis name

Property: Action

Possible values

- Compute
- DLL-Call
- Dialog-Call
- Dialog Accept
- Dialog Cancel

Property for

- PushButton

Definition

The action carried out when a pushbutton is clicked.

The only way to close secondary dialogs and finish the user interaction between the secondary dialog and the user is with the following predefined logical events:

- Logical event: Accept. Every data manipulation (in general every user interaction) that happened from the time when the dialog was invoked is accepted. The dialog will be closed and dialog control goes back to the place in the underlying XPL-application from where the dialog was initiated. If the application property calculation; is set to automatic, the XPL-application is computed (intercommunication with the application model). If the application property calculation is set to manual, the XPL-application is not computed only checked.
- Logical event: Cancel. Every data manipulation that happened from the time when the dialog was invoked is canceled by the Designer Runtime. The dialog will be closed and dialog control goes back to the place in the underlying XPL-application from where the secondary dialog was initiated. The status of the XPL-application is exactly the same as before the secondary dialog was called. The logical events Accept and Cancel are mapped to the standard user interactions for secondary dialogs (keyboard Esc, the close symbol x in the window title and so on) as well as to specific Pushbutton actions (see below).

Example

Assume the following layout:



Clicking on the PushButton would open the dialog specified by property <Dialog description>.

Property: ActivColor

Possible values

- AppDef
- UserDef

Property for

- ComboBox
- EditField

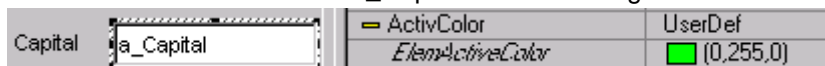
Definition

Determines whether the active color for an element is:

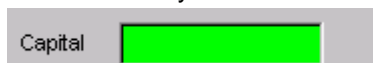
- AppDef: The default color for the application (see AppActiveColor).
- UserDef: A color defined by the user.

Example

The active color for EditField a_Capital is defined as green:



When the EntryField is selected, it turns green:



Property: Alignment

Possible values

- Left
- Right

Property for

- CheckBox . Type = CheckBox
- EditField . Type = Currency / Date / Mask / Number / Text
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Currency / Date / Mask / Number / Text
- GridList . Property for = Column 1 . Type = Currency / Number / Text

- Label . Type = Label
- RadioButton . Type = RadioButton
- RadioGroup . Type = RadioButton
- ResultField . Type = Currency result / Date result / Mask result / Number result / Result

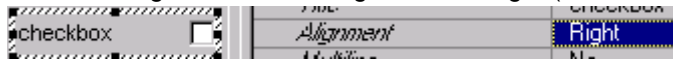
Definition

Alignment of the element.

Examples

Example 1

The following CheckBox is aligned to the right (the text is on the left).
The following CheckBox is aligned to the right (the text is on the left).

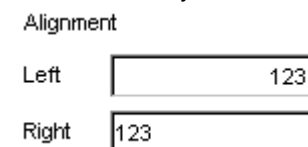


Example 2

The following layout has EditFields with `<Alignment> = <Right>`, `<Left>`.



The runtime layout:



Property: AppActiveColor

Possible values

- (0..255,0..255,0..255)

Property for

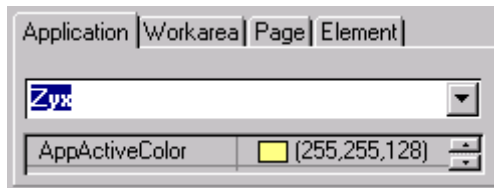
- Application

Definition

Active color of the application. This color is the default color for all elements (the color for a specific element can be defined using `ActivColor`).

Example

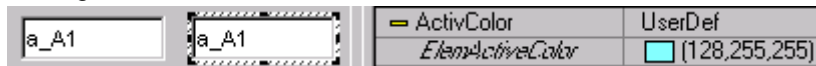
The active color for application Zyx is set to yellow:



The left EditField uses the active color defined for the application.



The right EditField has its own active color:



The following shows the results in the runtime layout:



Property: Attribute

Possible values

- An attribute from the model.

Property for

- Border . Type = Border
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = ComboBox
- GridList . Iteration type = Attribute
- GridList . Property for = Column 1 . Usage = Input
- Label . Type = Label
- Multimedia . Type = Media

Definition

Defines the model attribute for the layout component.

Example

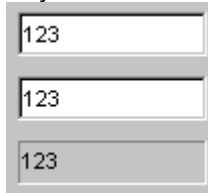
Assume the following layout with 2 EditFields and a ResultField, each with <Source> = <a_Variable1>.



In the runtime layout: Entering data in the top EditField will cause the following:

- The data is written to model attribute "a_Variable1".

- Any elements in the layout with `<Source> = <a_Variable1>` are updated with the new value.



Property: AVAILABLE-Property

Possible values

- A list of properties (separated by semicolons (;)).

Property for

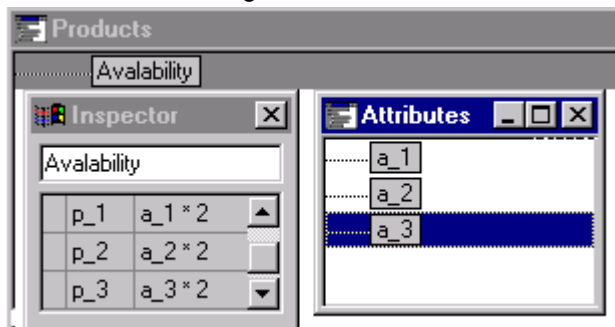
- Page

Definition

For any element on the page with `<ChkOwnAvailibility> = <Page>`: The element will only be available if the element is required for computing any of the result properties listed in `<AVAILABLE-Property>`.

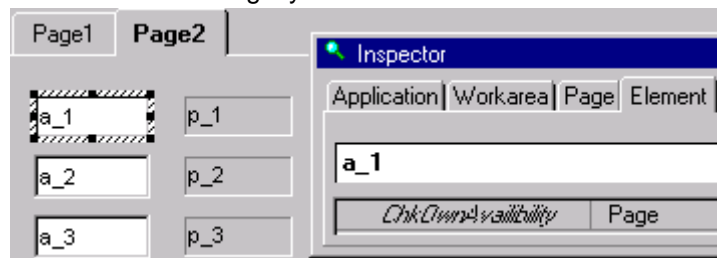
Example

Assume the following model:

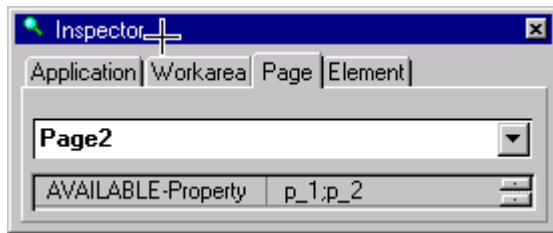


where `a_1`, `a_2`, `a_3` have default values 1, 2, 3.

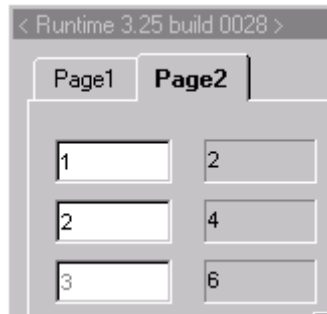
Assume the following layout:



EntryFields for model attributes `a_1`, `a_2`, `a_3` have element property `<ChkOwnAvailibility> = <Page>`. The page property `<AVAILABLE-Property> = <p_1;p_2>` (does not include `p_3`).



This would result in the following runtime layout:



Note that the EntryField for a_3 is not available (is grey).

Property: Background

Possible values

- (0..255,0..255,0..255)

Property for

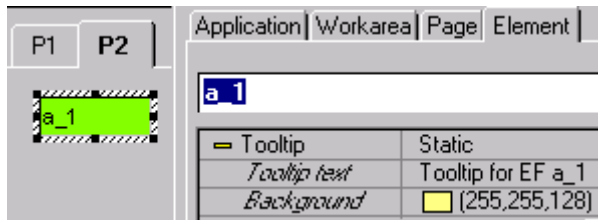
- CheckBox . Tooltip = Dynamic / Static
- ComboBox . Tooltip = Dynamic / Static
- EditField . Tooltip = Dynamic / Static
- ExtendedGridList . Tooltip = Yes
- Graphics . Tooltip = Yes
- GridList . Tooltip = Yes
- Label . Tooltip = Dynamic / Static
- Multimedia . Tooltip = Dynamic / Static
- Page . Tooltip = Yes
- PushButton . Tooltip = Dynamic / Static
- RadioButton . Tooltip = Dynamic / Static
- RadioGroup . Tooltip = Dynamic / Static
- ResultField . Tooltip = Yes
- Workarea . Tooltip = Yes

Definition

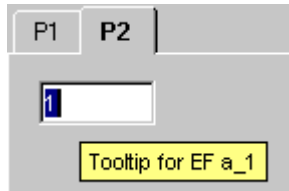
The background color.

Example

Assume a layout with EntryFields with a Static tooltip defined as shown:



The runtime layout tooltip would have a yellow background color:



Property: Background color(R,G,B)

Possible values

- (0..255,0..255,0..255)

Property for

- Application . Header = Yes
- Application . Footer = Yes
- Workarea . Header = Yes
- Workarea . Footer = Yes
- Line . Type = Line . Line color
- 3DBorder . Type = 3D-Area
- Border . Type = Border
- Label . Type = Label
- Graphics
- Page
- RadioGroup
- ResultField
- CheckBox
- ComboBox
- EditField
- ExtendedGridList . Property for = Column 1
- GridList . Property for = Column 1
- PushButton
- RadioButton

Definition

Color of the background.

See also:

- Property Background
- Default background color settings for Page.

Examples

Example 1

Assume a layout with an EditField with setting as shown:

Assume a layout with an EditField with setting as shown:

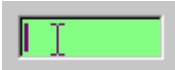
	Background color(R,G,B)	 (255,255,128)
	ActivColor	UserDef
	ElemActiveColor	 (128,255,128)

The runtime layout:

- EditField not selected:

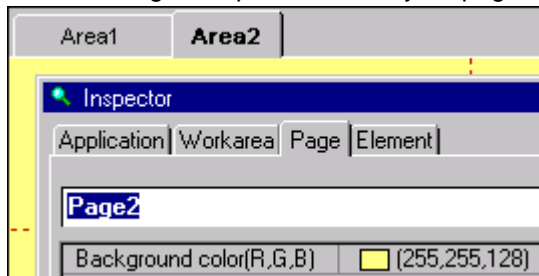


- EditField selected:



Example 2

The following example shows a layout page with a defined background color.



Property: Bounds

Possible values

- [0.. (top),0.. (left),0.. (width),0.. (height)]

Property for

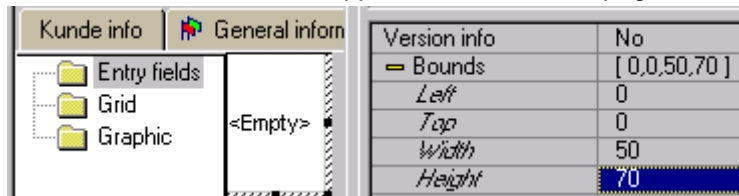
- Border
- Label
- Multimedia
- ResultField
- Line
- 3DBorder
- ExtendedGridList
- GridList
- CheckBox
- ComboBox
- EditField
- Graphics
- PushButton
- RadioButton
- RadioGroup

Definition

The location (top, left) and dimensions (width, height) of the element.

Example

The following example shows an entry field located in the uppermost left corner (0, 0). Note that the location is in relation to the upper left corner of the page not including the frame.



Property: Calculation

Possible values

- Automatic
- Manual

Property for

- Application

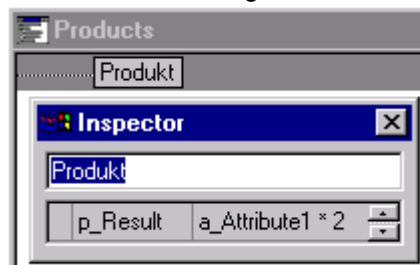
Definition

If Automatic: All elements on a page are re-calculated whenever a different page is selected.

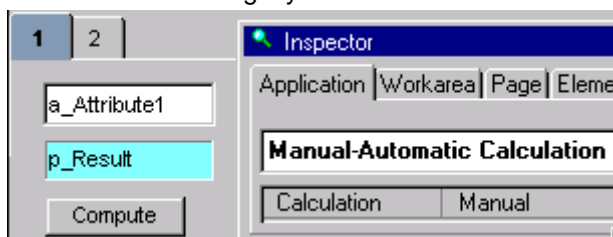
If Manual: Results are only computed when a PushButton is used to manually re-calculate.

Example

Assume the following model:



Assume the following layout:

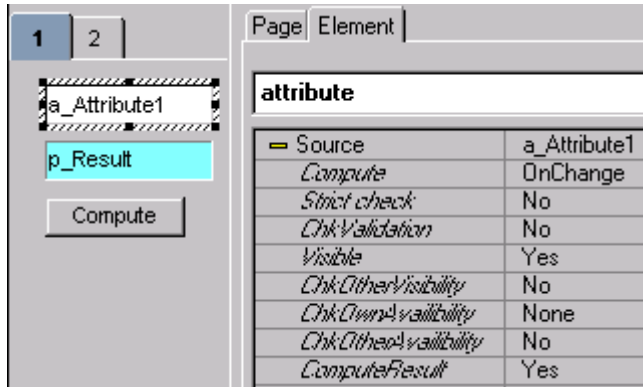


The PushButton <Action> = <Compute>:



The EntryField properties:

- <ComputeResult> = <Yes>
- <Compute> = <OnChange>



In the runtime layout, the result would not be calculated even if the pages were changed:



Pressing the PushButton would cause the result to be computed.



Note: The PushButton can be located anywhere in the layout (on any page or workarea).

Property: Caption Delete

Possible values

- Any text.

Property for

- ExtendedGridList
- Page . Multiple = Yes

Definition

The caption that will be displayed in the context menu (right-click) for deleting a member of an array. See also Caption New.

Example

The following example shows a multiple page with the Caption Delete set to "delete1P".

General info	Product info	Multiple	Yes
<ul style="list-style-type: none"> All products <ul style="list-style-type: none"> Product1 <ul style="list-style-type: none"> Product1P 		Identification	PageProduct1PName
		Count attribute	a_Persons
		Caption New	new1P
		Caption Delete	delete1P

In the layout test, right-clicking on an array member will bring up a context menu with the text "delete1P":



Property: Caption New

Possible values

- Any text.

Property for

- ExtendedGridList
- Page . Multiple = Yes

Definition

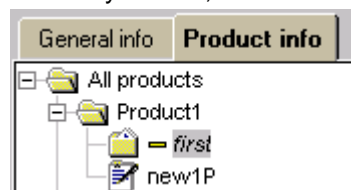
The caption that will be displayed in the layout for adding a member to an array.
See also Caption Delete.

Example

The following example shows a multiple page with the Caption New set to "new1P".

General info	Product info	Multiple	Yes
<ul style="list-style-type: none"> All products <ul style="list-style-type: none"> Product1 <ul style="list-style-type: none"> Product1P 		Identification	PageProduct1PName
		Count attribute	a_Persons
		Caption New	new1P
		Caption Delete	delete1P

In the layout test, the text will be displayed beside the icon for adding a new member to the array.



Property: ChkDynamic

Possible values

- Yes
- No

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup

Definition

If <ChkDynamic> = <Yes>: The following occurs:

- The model property <default> is re-evaluated for all model attributes.
- For all layout Labels with property <Attribute> = (a model attribute): The corresponding model attribute property <label> is re-evaluated and assigned to the label.

When exactly the above occurs is determined by:

- If <Compute> = <OnChange>: When data (a digit or letter) is entered in the EntryField.
- If <Compute> = <OnExit>: When the focus changes after data (a digit or letter) has been entered in the EntryField.

See also: Setting the default value for ChkDynamic.

Example

See the discussion of re-computing attribute default / label.

Property: ChkOtherAvailibility

Possible values

- Yes
- No

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup

Definition

Determines whether or not the availability of the other elements is to be determined when the content of the element is changed.

See also:

- Setting the default value.
- Concepts: Availability

Property: ChkOtherVisibility

Possible values

- No
- Yes

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup

Definition

If <Yes>: Changing the value in the element will cause the visible property for all elements in the page with dynamic visibility to be checked (reevaluated).

See also:

- Concept: Dynamic visibility.
- Setting the default value.

Property: ChkOwnAvailability

Possible values

- None
- All
- List
- Page

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- PushButton
- Label
- Border
- RadioGroup

Definition

Specifies a set of result elements. The element is available if the element is required for the computation of at least 1 of the result elements in the set. If <ChkOwnAvailability> =

- <None>: Element is ALWAYS available.
- <All>: Element is available IF the element's model attribute is required for ANY result on ANY page.
- <List>: Element is available IF the element's model attribute is required for ANY result in the list of results specified by element property <List (...;...)> (the listed results can be on any page).
- <Page>: Element is available IF the element's model attribute is required for ANY result in the list of results specified by page property <AVAILABLE-Property> (the listed results can be on any page).

See also:

- Setting the default value.

Example

For an example see Results list.

Property: ChkValidation

Possible values

- Yes
- No

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup

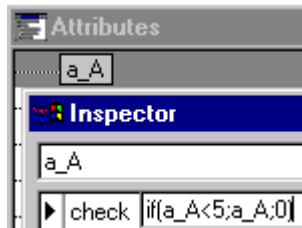
Definition

If <Yes>: When a value is entered (or selected): The equation defined (in the model) for the element attribute property <Check> will be executed. This equation is typically an IF statement that returns 1 of 2 possible values depending on the value entered. The returned value will be displayed in the element. Typically 1 of the returned values is the attribute itself.

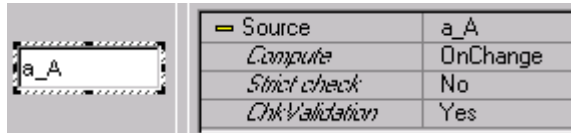
See also: Setting the default value.

Example

Assume the following model attribute:



Assume the following layout EditField:



When the runtime layout is opened, the value of 0 is displayed since a_1 has no default value (is null).



Clicking on the key "5" would result in 0 again being displayed.

Clicking on the key "4" would result in 4 being displayed.

Clicking on the key "4" again would result in 0 being displayed, since the entered value was 44.

Property: Color

Possible values

- (0..255,0..255,0..255)

Property for

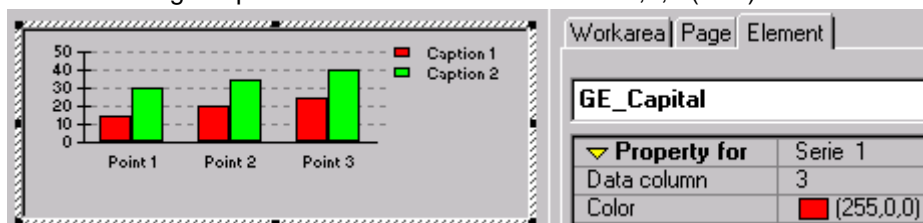
- Graphics . Property for = Serie 1

Definition

Color of the series.

Example

In the following Graphic the <Color> for Serie 1 is 255,0,0 (Red):



Property: Column

Possible values

- Positive Integer

Property for

- ExtendedGridList
- GridList

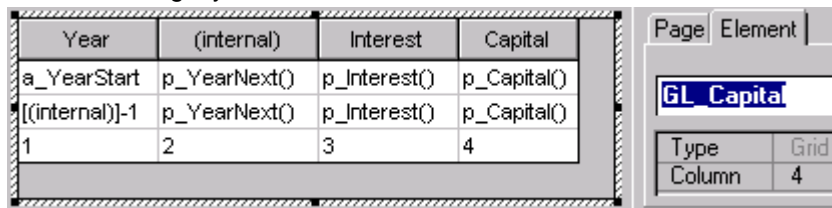
Definition

The number of columns in a Grid / ExtendedGrid.

Example

In the following layout the Grid has 4 columns:

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4



Note: Normally the Grid / ExtendedGrid next-key column column has <Width> = 0 (ie, the column is not visible).

Property: Columns moveable

Possible values

- Yes
- No

Property for

- ExtendedGridList
- GridList

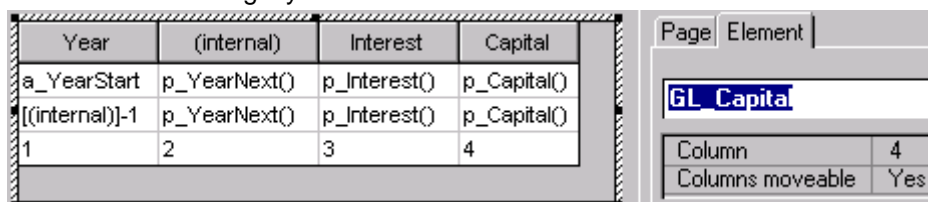
Definition

If <Yes>: The column is "moveable" (can be resized).

Example

Assume the following layout with a Grid with <Columns moveable> = <Yes>.

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4



In the runtime layout the columns can be resized:

Y	(int	Inte	Capital
1	2	33	133 DM
2	3	43	176 DM
3	4	58	235 DM

Note: The total width of the columns cannot exceed the width of the element. For the above Grid the maximum width of the element is shown below:

Year	(int	Inte	Capital
1	2	33	133 DM
2	3	43	176 DM
3	4	58	235 DM

Property: Compute

Possible values

- OnExit
- OnChange.

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup

Definition

If <ComputeResult> = <Yes> and <Calculation> = <Automatic>: Model product results whose calculations include the the model value represented by the element are re-calculated based on the setting of <Compute>.

See also:

- Concept: Compute results.
- Setting the default value for elements (Compute event).

Property: ComputeResult

Possible values

- Yes
- No

Property for

- EditField
- ComboBox
- CheckBox

- RadioButton
- RadioGroup

Definition

If <ComputeResult> = <Yes> and <Calculation> = <Automatic>: Model product results whose calculations include the the model value represented by the element are re-calculated based on the setting of <Compute>.

See also:

- Concept: Compute results
- Setting the default value.

Property: Conversion List

Possible values

- Name of a model attribute

Property for

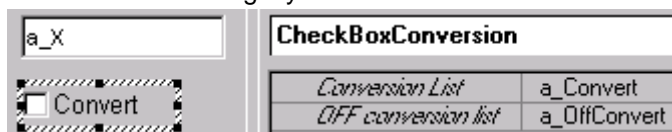
- CheckBox . Type = CheckBox
- RadioButton . Type = RadioButton

Definition

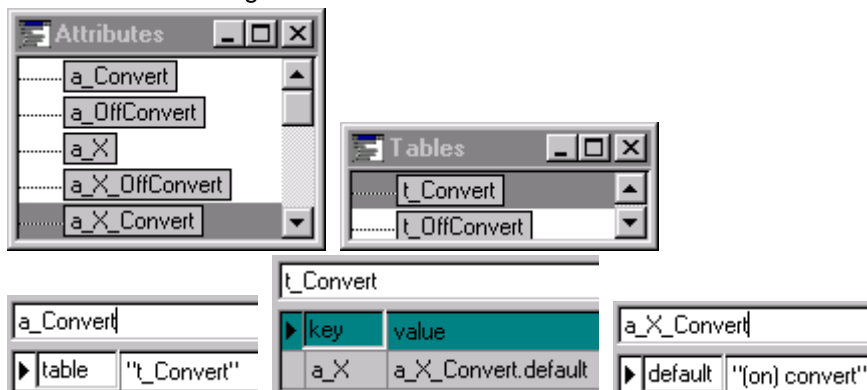
Specifies the model attribute whose property <table> references the (ON) conversion table when the element is selected.

Example

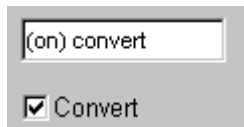
Assume the following layout:



Assume the following model:



In the runtime model, the attribute a_X would be converted to a_X_Convert.default when the CheckBox is checked:



Property: ConvertAttr

Possible values

- Name of a model attribute

Property for

- PushButton

Definition

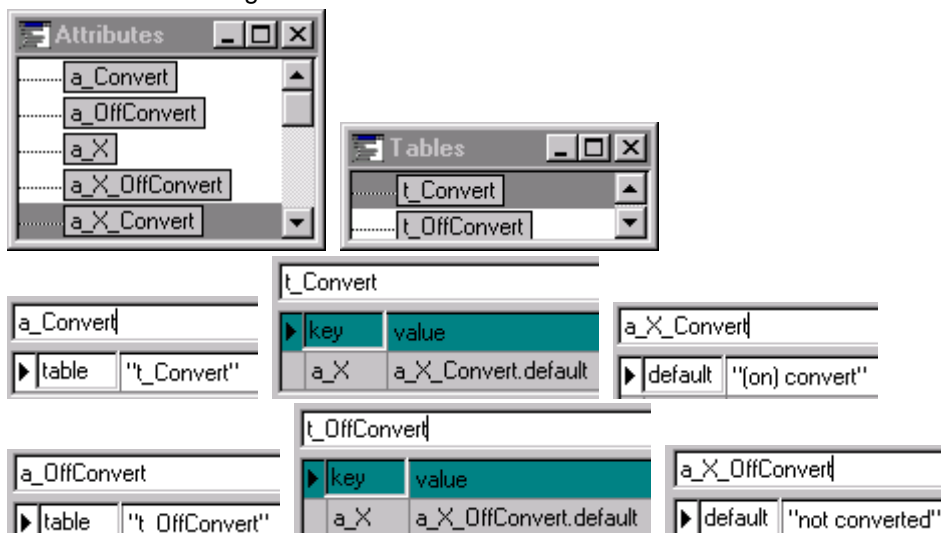
Specifies the model attribute whose property <table> references the conversion table when the PushButton is clicked.

Example

Assume the following layout:

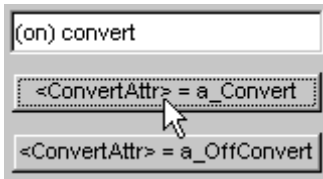


Assume the following model:

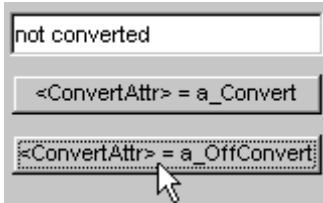


In the runtime layout:

- Attribute a_X would be converted to a_X_OffConvert.default when the top PushButton is clicked.



- Attribute a_X would be converted to a_X_OnConvert.default when the bottom PushButton is clicked.



Property: Count attribute

Possible values

- Name of model attribute

Property for

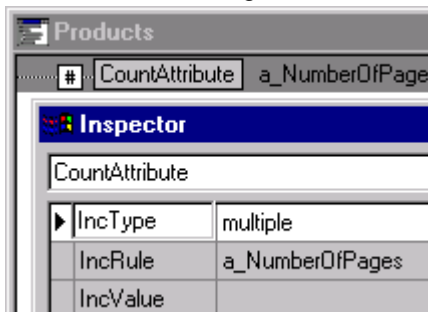
- ExtendedGridList
- Page . Multiple = Yes

Definition

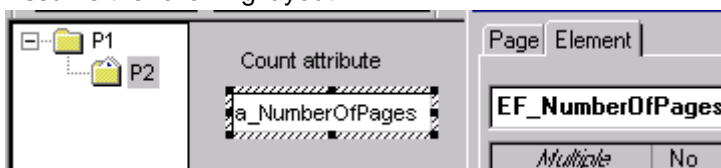
The model attribute that stores the number of multiple pages (Page) or number of rows (ExtendedGrid).

Example

Assume the following model:

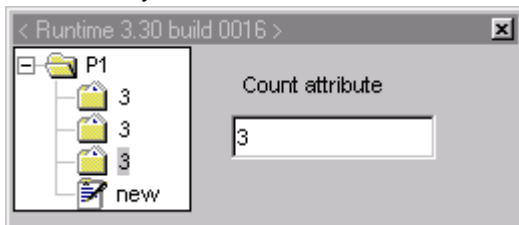


Assume the following layout:



Page Element	
P_Product11	
Multiple	Yes
Identification	EF_NumberOfPages
Count attribute	a_NumberOfPages
Caption New	new
Caption Delete	delete

Runtime Layout with 3 subnodes:



Property: Currency symbol

Possible values

- On
- Off

Property for

- EditField . Type = Currency
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Currency
- ResultField . Type = Currency result
- GridList . Property for = Column 1 . Type = Currency

Definition

If <On>: A currency symbol will be displayed.

Example

Assume the following layout:

Text	<input type="text" value="a_variable1"/>	<table border="1"> <thead> <tr> <th colspan="2">EditFieldNoSymbol</th> </tr> </thead> <tbody> <tr> <td>Type</td> <td>Currency</td> </tr> <tr> <td>Minimum</td> <td>0</td> </tr> <tr> <td>Maximum</td> <td>0</td> </tr> <tr> <td>Currency symbol</td> <td>On</td> </tr> <tr> <td>Decimal position</td> <td>0</td> </tr> <tr> <td>Alignment</td> <td>Right</td> </tr> </tbody> </table>	EditFieldNoSymbol		Type	Currency	Minimum	0	Maximum	0	Currency symbol	On	Decimal position	0	Alignment	Right
EditFieldNoSymbol																
Type	Currency															
Minimum	0															
Maximum	0															
Currency symbol	On															
Decimal position	0															
Alignment	Right															
Currency (with symbol)	<input type="text" value="DM"/>															
Currency (without symbol)	<input type="text"/>															

All 3 of the above EditFields have the same <Source> attribute. The second EditField has <Currency symbol> = <Yes>.

In the runtime layout:

Text	123,45
Currency (with symbol)	123 DM
Currency (without symbol)	123

Property: Data column

Possible values

- Integer

Property for

- Graphics . Property for = Serie 1

Definition

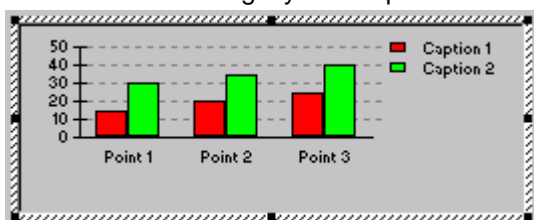
Specifies the source Grid / ExtendedGrid data column for the Serie.

Example

Assume the following layout Grid:

Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4

Assume the following layout Graphic:



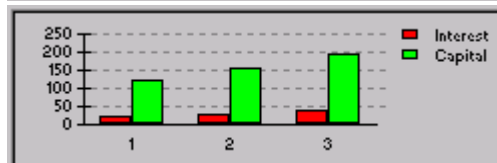
▼ Property for	Serie 1
Data column	3
Color	■ (255,0,0)
▼ Property for	Serie 2
Data column	4
Color	■ (0,255,0)

Model product properties will be shown in series:

- p_Interest() : serie 1 (red)
- p_Capital() : serie 2 (green)

In the runtime layout:

Year	(internal)	Interest	Capital
1	2	25 DM	125 DM
2	3	31 DM	156 DM
3	4	39 DM	195 DM



Property: Data elements

Possible values

- Selected
- Not selected

Property for

- Page

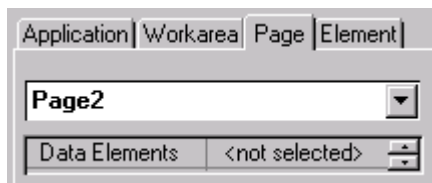
Definition

If <Selected>: Data elements have been selected for the page.

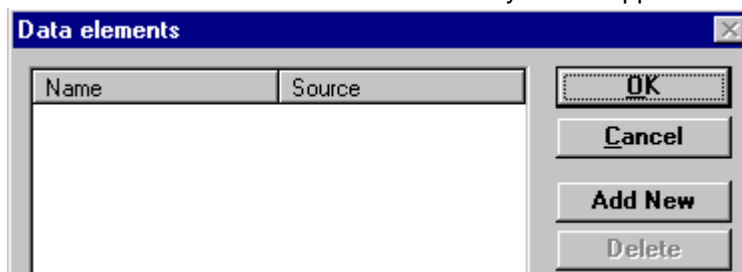
Data elements provide VFrame with access to model attributes that are not included as elements in the layout.

Example

Assume that no data elements have been selected for the page:



Click on the <not selected> text. A directory button appears. Click on the button.



Click "Add New". A new element (without a model attribute) is added to the list.

Name	Source
Element43403D	<Empty>

Change the name (optional). Select the Source.

Name	Source
Variable1	a_Variable1

Click "OK". Note that <Data elements> = <selected>.

Data Elements	<selected>
---------------	------------

Property: DataIndicator

Possible values

- On
- Off

Property for

- Page
- Workarea

Definition

Specifies if a data indicator should be displayed.

Example

In the following layout, the data indicator for the page is on:



Property: Decimal position

Possible values

- Non-negative Integer

Property for

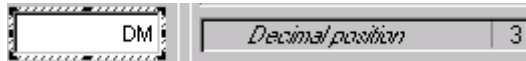
- EditField . Type = Currency / Number
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Currency / Number
- ResultField . Type = Currency result / Number result
- GridList . Property for = Column 1 . Type = Currency / Number

Definition

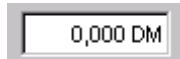
Number of places after the decimal point.

Example

Assuming the following layout:



In the runtime layout:



When entering data, only a maximum of 3 digits after the decimal point can be entered:



Property: Default font

Possible values

- A font / fontsize (other font properties can also be selected but are not shown in the Inspector)

Property for

- Application
- Border . Type = Border
- CheckBox
- ComboBox
- EditField
- Label . Type = Label
- PushButton
- RadioButton
- ResultField

Definition

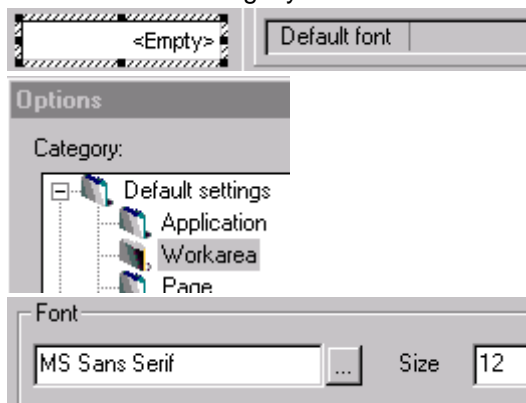
The default font.

See also:

- Setting the default Application default font.
- Setting the default Workarea default font.

Example

Assume the following layout:



In the runtime layout the EditField contents would use the default font/size specified above:

abc123def456

Click on the <Default font> value field. A button appears. Click on the button. The "font style" dialog appears. Select the font style and characteristics:



Click "OK". Note the <Default font> property:



In the runtime layout:

123abc45

Property: Defolding

Possible values

- Automatic
- Manual

Property for

- Page

Definition

If <Automatic>: A visible node's immediate subnodes are displayed when the workarea is displayed.

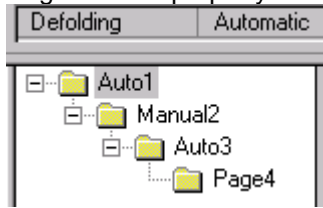
If <Manual>: A visible node's immediate subnodes are only displayed when the node icon or text is double-clicked.

Example

Assume the following layout with:

- Page "Auto1" property <Defolding> = <Automatic>
- Page "Manual2" property <Defolding> = <Manual>

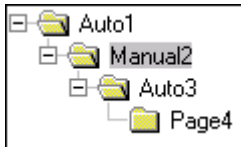
- Page "Auto3" property <Defolding> = <Automatic>



Upon opening the runtime layout:



Double-clicking on "Manual2" yields:



Changing the workareas does not affect the current state of the page tree.

Double-clicking on "Auto1" folds the tree. Double-clicking on "Auto1" again opens the subtree with the unfolded state for the subtree for "Manual2" unchanged.

Property: Devisualize

Possible values

- Selected
- Not selected

Property for

- CheckBox . Type = CheckBox
- RadioButton . Type = RadioButton

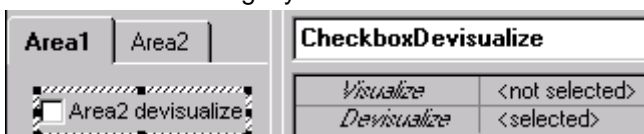
Definition

If <Selected>: A page / workarea has been selected as being "devisualized" when the element is selected.

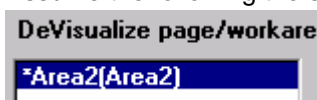
If <Not selected>: Selecting the element has no effect.

Example

Assume the following layout:



Assume the following the selected pages/workareas for devisualization:



The runtime layout:



Property: DI Rule

Possible values

- Model attribute / property that evaluates to boolean.

Property for

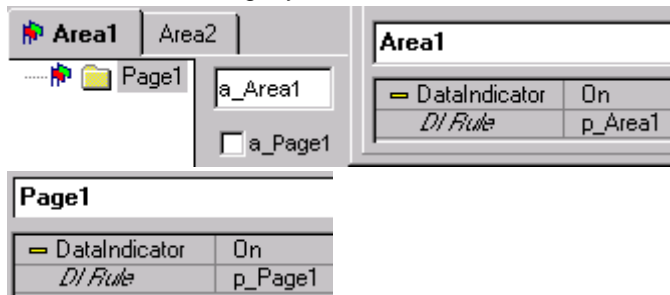
- Page . DataIndicator = On
- Workarea . DataIndicator = On

Definition

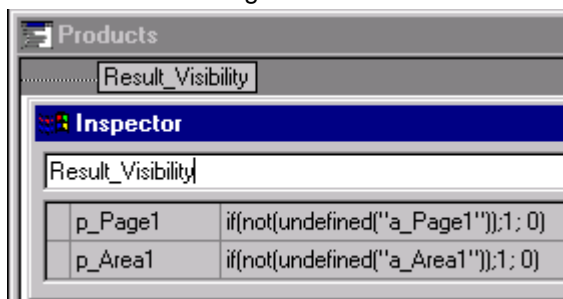
The rule that determines when the data required / data not required icons will be displayed for the data indicator.

Example

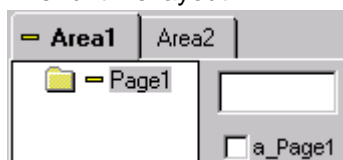
Assume the following layout:

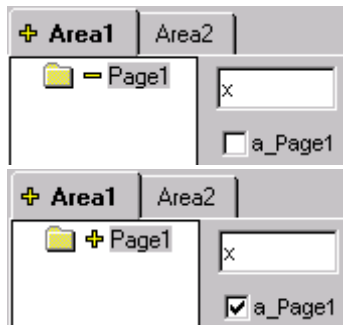


Assume the following model:



The runtime layout:





Property: Dialog-Description

Possible values

- Directory / filename of runtime layout (.vpc)

Property for

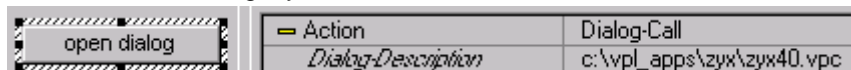
- PushButton . Action = Dialog-Call

Definition

Specifies the runtime layout to be opened as a subdialog when the PushButton is clicked.

Example

Assume the following layout element:



In the runtime dialog, clicking on the PushButton would open the runtime layout [zyx40.vpc](#) in a subdialog.



Property: Dll-Description

Possible values

- Path and name of the dll description .ini file

Property for

- PushButton . Action = DLL-Call

Definition

Specifies the directory and filename of the configuration file (.ini) for the dll that is executed when the PushButton is clicked.

Example

Assume the following dialog:

dll call	Action	DLL-Call
	<i>Dll-Description</i>	c:\vpl_apps\zyx\zyx41_kndneu.ini

In the runtime layout, when the PushButton is clicked a call to the .dll whose interface is specified in zyx41_kndneu.ini is made.

Property: ElemActiveColor

Possible values

- (0..255,0..255,0..255)

Property for


- ComboBox . ActivColor = UserDef
- EditField . ActivColor = UserDef

Definition

Color of the active element.

Example

Assume the following layout element:

<Empty>	ActivColor	UserDef
	<i>ElemActiveColor</i>	 (0,255,128)

In the runtime layout:

- Element does not have the focus



- Element has the focus



Property: Element3D

Possible values

- Yes
- No

Property for

- CheckBox
- ComboBox
- EditField
- Graphics
- PushButton
- RadioButton
- RadioGroup
- ResultField

Definition

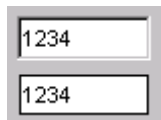
If Yes: The element is 3 dimensional.

Example

Assume the following layout:



In the runtime layout:



Property: Font

Possible values

- A font / fontsize (other font properties can also be selected but are not shown in the Inspector).

Property for

- CheckBox . Tooltip = Dynamic / Static
- ComboBox . Tooltip = Dynamic / Static
- EditField . Tooltip = Dynamic / Static
- ExtendedGridList . Property for = Column 1
- ExtendedGridList . Tooltip = Yes
- Graphics
- Graphics . Tooltip = Yes
- GridList . Property for = Column 1
- GridList . Tooltip = Yes
- Label . Tooltip = Dynamic / Static
- Multimedia . Tooltip = Dynamic / Static
- Page . Tooltip = Yes
- PushButton . Tooltip = Dynamic / Static
- RadioButton . Tooltip = Dynamic / Static
- RadioGroup . Tooltip = Dynamic / Static
- ResultField . Tooltip = Yes

- Workarea . Tooltip = Yes

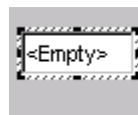

Definition

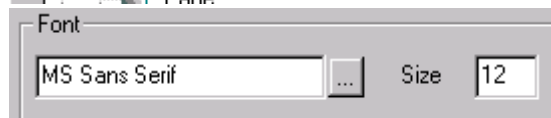
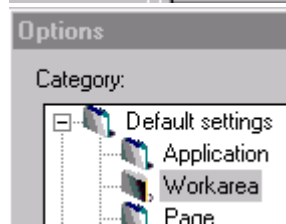
The default font.

See also: Setting the default Application default font.

Example

Assume the following layout:

	<ul style="list-style-type: none"> Tooltip 	Static
	<i>Tooltip text</i>	tooltip text
	<i>Background</i>	 (255,255,128)
	<i>Font</i>	





In the runtime layout the EditField tooltip would use the default font/size specified above:



Click on the value field. A button appears. Click on the button. The "font style" dialog appears. Select the font style and characteristics:



Click "OK". Note the property:

	<ul style="list-style-type: none"> Tooltip 	Static
	<i>Tooltip text</i>	tooltip text
	<i>Background</i>	 (255,255,128)
	<i>Font</i>	Eras Bold ITC , 10

In the runtime layout:



Property: Footer

Possible values

- No
- Yes

Property for

- Application
- Workarea

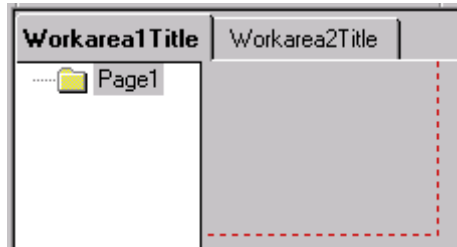
Definition

If <Yes>: A footer is added.

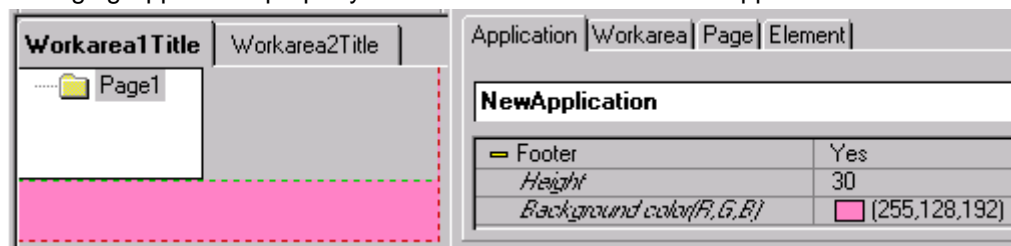
Note: If the footer contains no elements: If the layout is saved: The footer is deleted.

Example

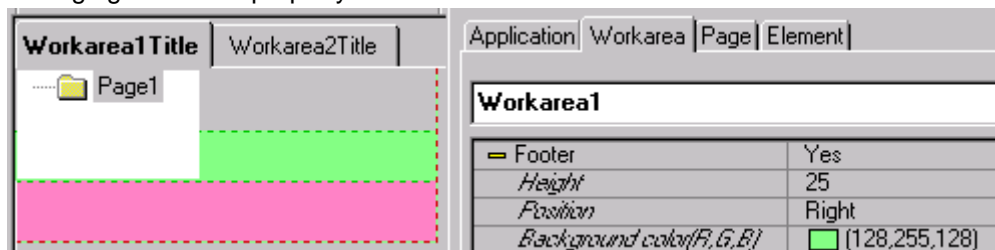
Assume the following layout:



Changing Application property <Footer> to <Yes> causes an application footer to be added.



Changing Workarea property <Footer> to <Yes> causes a workarea footer to be added.



Property: Format (element)

Possible values

- dd/mm/yy
- dd/mm/yyyy
- mm/dd/yy
- yy/mm/dd
- dd.mm.yyyy
- dd-mm-yyyy
- MMM dd,yyyy
- dd-MMM-yy

Property for

- EditField . Type = Date
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Date
- ResultField . Type = Date result

Definition

The format (selected from a drop-down list) of the date in the element.

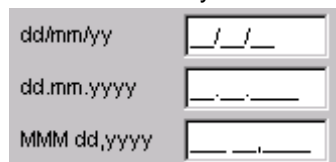
Example

Assume the following layout:

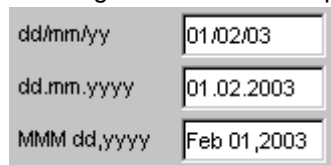


Assume that the 3 above EditFields have the same <Source>.

In the runtime layout:



Entering "010203" in the top EditField results in the following:



Note: Invalid input is ignored.

Property: Format (application)

Possible values

- Individual
- 640*480
- 800*600
- 1024*768

Property for

- Application

Definition

The format of the application.

Different screen resolutions should be supported by the designer and its runtime.

The appearance of the user interface according to the different screen resolutions is totally defined at design time for three resolutions:

- Screen resolution: 640 * 480 Pixels
- Screen resolution: 800 * 600 Pixels
- Screen resolution 3: 1024 * 768 Pixels

In this case, the first design of the user interface will be done for one of the explicitly previously selected screen resolutions (new application property Screen resolution) the so called primary resolution.

After finishing the GUI-design for the primary resolution, the other layouts (the so called secondary resolutions) can be generated automatically by the VPMS designer (activated through new menu items).

The basic principles of this automatic generation can be influenced previously to the generation process by individually determining the appropriate font specifications for each screen resolution (the user can maintain a font map with two rows, the first row refers to source font and the second one contains the corresponding font for target resolution.)

Generating the layouts for the secondary resolutions is done by the VPMS Designer by changing the font for each object (using the font map as lookup information) and repositioning the objects on the screen (recalculating is done by following a proportional factor, that can be either manually defined for each resolution or is determined automatically by the VPMS Designer). Generating the layout for the secondary resolution will cause appearance of a new application open in Designer.

To store the new layout definition with keeping the primary resolution layout a menu option Save as should be used to store the new layout in a new VPL-file.

The different layouts for the screen resolutions can be maintained by the user manually, after they have been generated once. This can be done by switching between the layout definitions within the VPMS Designer (new menu items and actions will be required for this feature, e.g. a dropdown list box in the toolbar with three selectable entries 640*480, 800*600 and 1024 * 768).

To assist the user by keeping the different layouts consistent if one of the layouts was changed, the VPMS Designer has a check synchronicity action, that lists all the possible differences between the primary resolution layout and the secondary resolution layouts in a non-modal report window (dialog box). By following this dialog box, the user can update the layouts of the secondary resolutions manually and keep them identical with the layout of the primary resolution.

At runtime, it is the part of the frame application to pass a parameter to the Designer runtime which resolution layout should be used.

The advantage of this alternative is, that almost no process overhead is required at runtime to support the different screen resolutions (the controls and the whole interface requires no ongoing and time consuming recalculation). The performance impacts will be for seeable minor.

Another advantage is, that the user can define for each screen resolution a total different user interface for the application also (e.g. information, that is spread over several notepages in the 640 * 480 resolution can be combined in a single notepage in the other resolutions).

Property: Frame font

Possible values

- A font / fontsize (other font properties can also be selected but are not shown in the Inspector).

Property for

- Workarea . Style = Frame

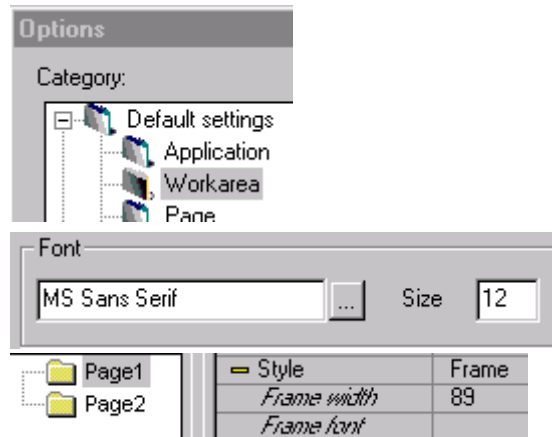
Definition

The default font for the frame text.

See also: Setting the default Workarea default font.

Example

Assume the following layout with the default font for the frame text:



Click on the <Frame font> value field. A button appears. Click on the button. The "font style" dialog appears. Select the font style and characteristics:



Click "OK". Note the <Frame font> property (the color is not displayed):



In the runtime layout:



Property: Frame width

Possible values

- Non-negative Integer

Property for

- Workarea . Style = Frame

Definition

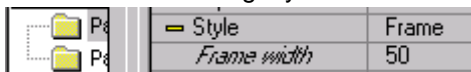
The width (in pixels) of a frame in a workarea (workarea with style specified as frame).

Note: The width can also be set by moving the right edge of the frame with the mouse.

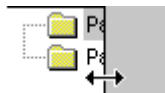
See also: Setting the default Frame width.

Example

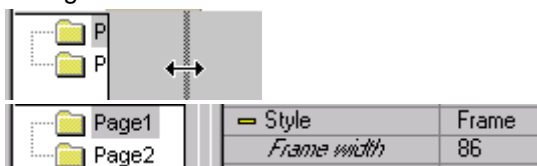
Assume the following layout with a frame:



Place the cursor over the right edge of the frame. The cursor changes to a double arrow.



Press and hold the left mouse button. Drag the mouse and then release the left mouse button to change the frame width.



Property: FrameMoveable

Possible values

- No
- Yes

Property for

- Workarea . Style = Frame

Definition

Specifies if the frame in the runtime layout is "moveable" (ie, can be resized).

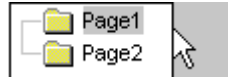
Note: The frame is always resizeable in the design-time layout.

Example

Assume the following layout:

Page1	Frame width	82
Page2	Frame font	
	FrameMoveable	No

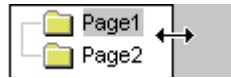
In the runtime layout: Note that the cursor does not change to a double arrow for resizing the frame.



Change <FrameMoveable> = <Yes>.

Page1	FrameMoveable	Yes
Page2	MinFrameWidth	0
	MaxFrameWidth	0

Note that the frame can now be resized in the runtime layout:



Property: Graphics type

Possible values

- Line
- Vertical Bar
- Horizontal Bar
- Vertical 3D bar
- Vertical Gantt
- Horizontal Gantt
- Pie
- 3D Pie

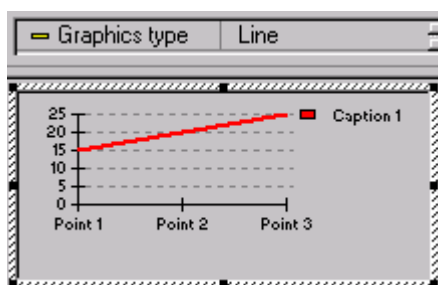
Property for

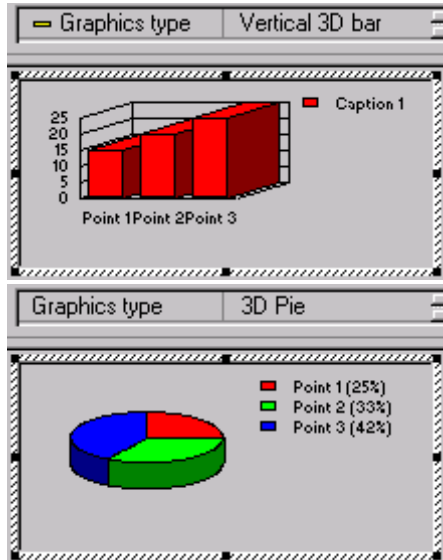
- Graphics

Definition

Specifies the graphics type.

Examples





Property: Group

Possible values

- Name of RadioButton group

Property for

- RadioButton . Type = RadioButton

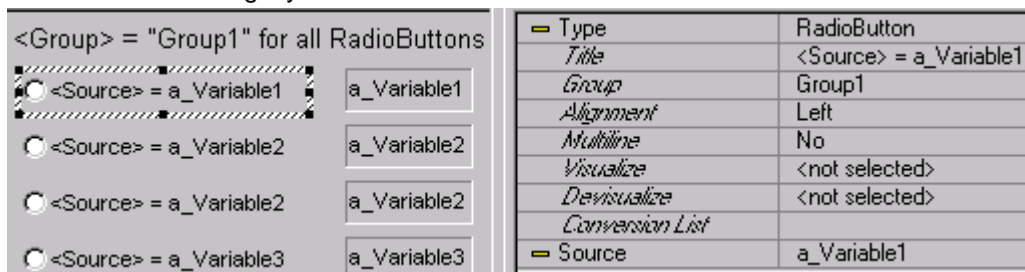
Definition

Defines the Group that the RadioButton belongs to. Only 1 RadioButton in a RadioGroup can be enabled at any time (assuming that the <Source> for each RadioButton is different).

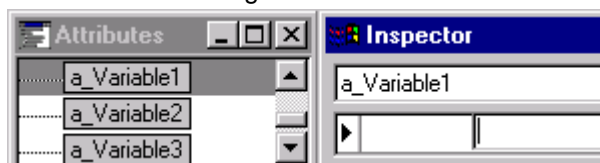
Note: A Group is created by entering the name of the Group in the <Group> drop-down list.

Example

Assume the following layout:



Assume the following model:



In the runtime model:

<Group> = "Group1" for all RadioButtons

<input checked="" type="radio"/> <Source> = a_Variable1	1
<input type="radio"/> <Source> = a_Variable2	0
<input type="radio"/> <Source> = a_Variable2	0
<input type="radio"/> <Source> = a_Variable3	0

Note that when selecting the second or third RadioButton, both buttons are selected, since both buttons reference the same <Source>.

<Group> = "Group1" for all RadioButtons

<input type="radio"/> <Source> = a_Variable1	0
<input checked="" type="radio"/> <Source> = a_Variable2	1
<input checked="" type="radio"/> <Source> = a_Variable2	1
<input type="radio"/> <Source> = a_Variable3	0

Property: Header

Possible values

- No
- Yes

Property for

- Application
- Workarea

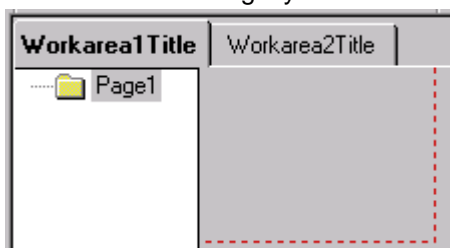
Definition

If <Yes>: A header is added.

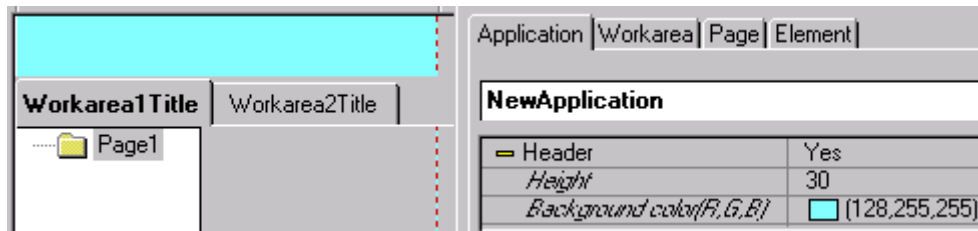
Note: If the Header contains no elements: If the layout is saved: The header is deleted.

Example

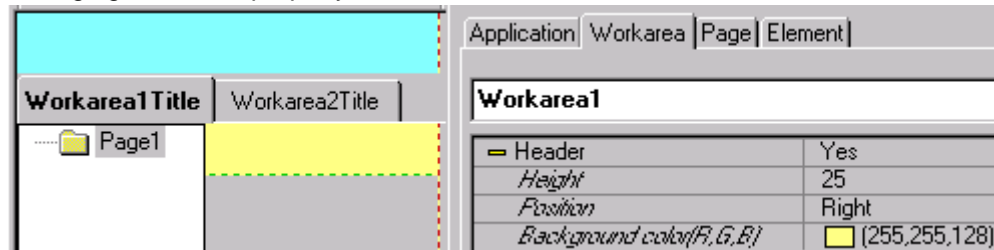
Assume the following layout:



Changing Application property <Header> to <Yes> causes an application header to be added.



Changing Workarea property <Header> to <Yes> causes a workarea header to be added.



Property: Header font

Possible values

- A font / fontsize (other font properties can also be selected but are not shown in the Inspector).

Property for

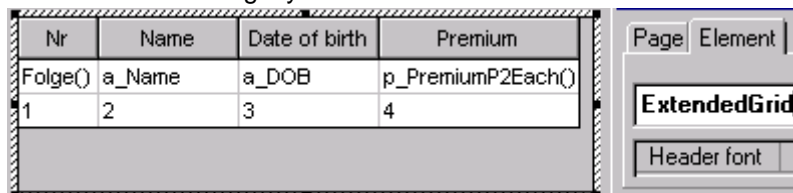
- GridList
- ExtendedGridList

Definition

Specifies the header font.

Example

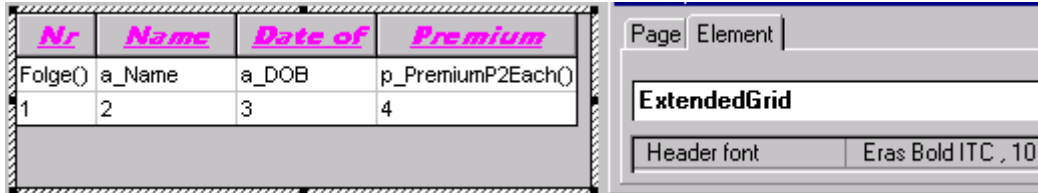
Assume the following layout ExtendedGrid:



Click on the <Header font> value field. A button appears. Click on the button. The "Font style" dialog appears. Select the font style and characteristics:



Click "OK". Note the property:



Property: Height (application)

Possible values

- Non-negative Integer.

Property for

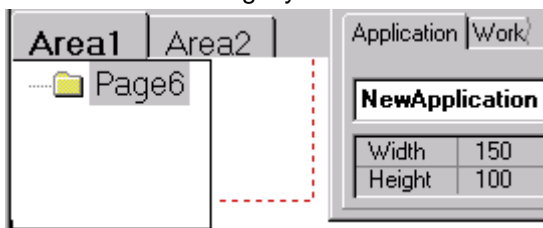
- Application

Definition

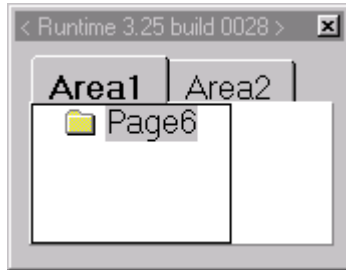
Specifies the height of the application.

Example

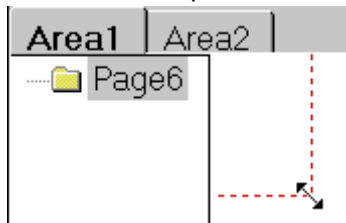
Assume the following layout:



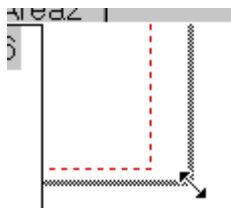
In the runtime layout:



The mouse change be used to change the Width/Height. Move the mouse over the red-dotted line so that the mouse pointer becomes a double-arrow.



Press and hold the left mouse button while moving the mouse to change the application width/height.



Property: Height (Header/Footer)

Possible values

- No
- Yes

Property for

- Application
- Workarea

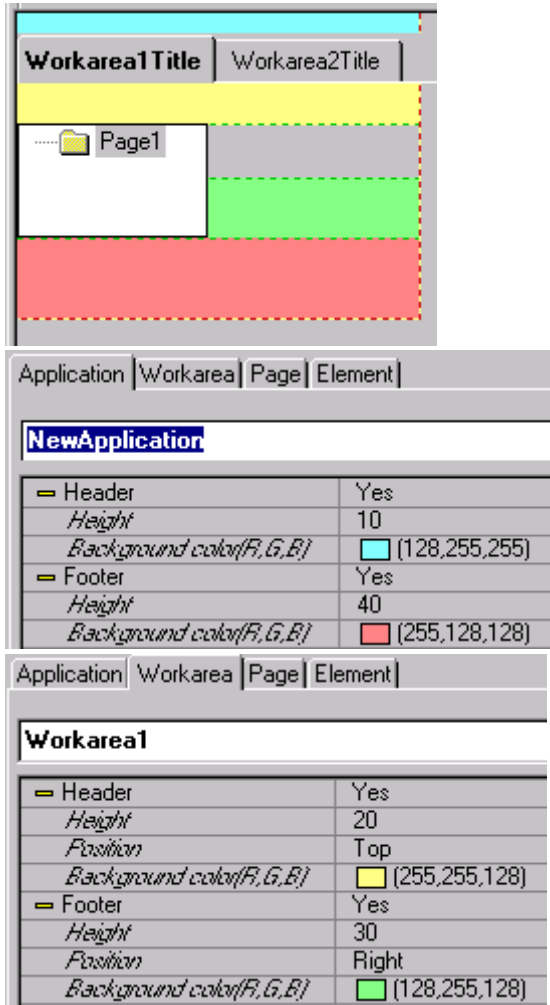
Definition

Specifies the height of the Header / Footer.

Example

The following layout has:

- Application header with <Height> = 10.
- Workarea header with <Height> = 20.
- Workarea footer with <Height> = 30.
- Application footer with <Height> = 40.



Property: HelpID

Possible values

- Integer.

Property for

- EditField
- ComboBox
- CheckBox
- RadioButton
- RadioGroup
- Graphics
- Page
- PushButton
- Resultfield
- Workarea

Definition

HelpID for the element.

Example

Assume the following layout EditField



Clicking "F1" in the runtime layout while the above EditField has the focus will cause the HelpID 123 to be sent to the the application help file.

Property: Identification

Possible values

- The name of an element on the page.

Property for

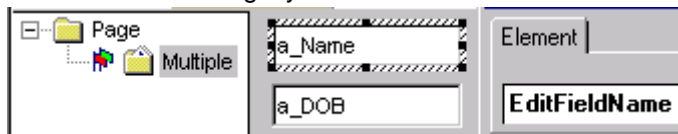
- Page . Multiple = Yes

Definition

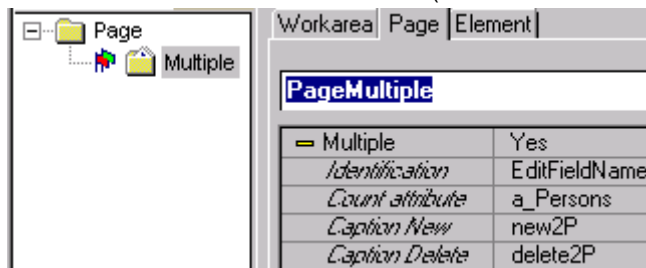
Identifies the element on the page that whose attribute value (a string) is the name of each of the multiple pages in the frame.

Example

Assume the following layout:



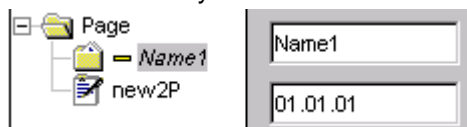
<Identification> = <EditFieldName> (the name of the EditField):



Note that the available values for <Identification> includes both EditFields.



In the runtime layout:



Property: Identification Grid

Possible values

- Specification of a grid.

Property for

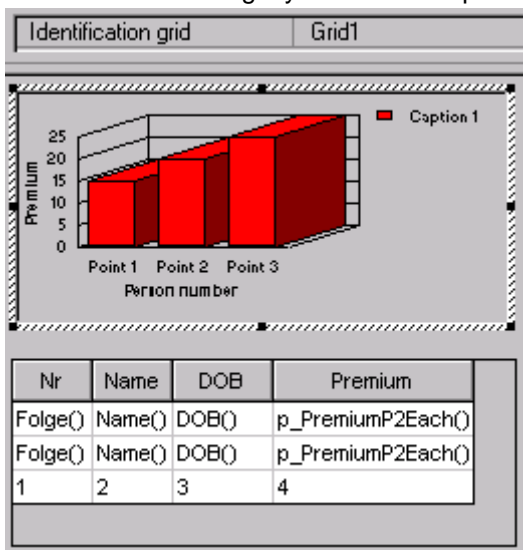
- Graphics

Definition

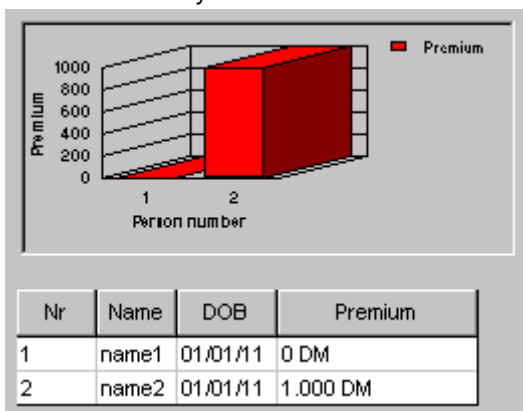
The name of the source grid for the graphic.

Example

Assume the following layout with a Graphics and the source Grid:



In the runtime layout



Property: Indexed

Possible values

- Yes
- No

Property for

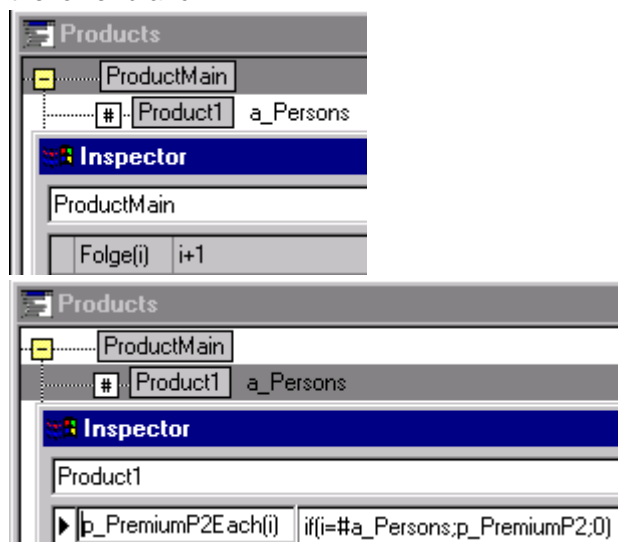
- IF (ExtendedGrid . Usage = Output) : ExtendedGrid . Property . Indexed

Definition

Specifies that the value to be angezeigt in an ExtendedGrid column is indexed in the model.

Example

Assume the following model with property <Folge(i)> in the upper branch and p_PremiumP2Each(i) in the lower branch.



The layout would require having <Zeilen-Indizierung> = <Ja> for columns that display the properties.

Nr	Name	Date of birth	Premium
Folge()	a_DOB	p_PremiumP2Each()	
1	3	4	

Application	Workarea	Page	Element
ExtendedGrid			
Property for	Column 1		
Title	Nr		
Width	40		
Usage	Output		
Type	Text		
Alignment	Left		
Property	Folge()		
Indexed	Yes		

Application Workarea Page Element	
ExtendedGrid	
▼ Property for	Column 4
Title	Premium
Width	100
Usage	Output
+ Type	Currency
- Property	p_PremiumP2Each()
<i>/indexed</i>	Yes

Property: Iteration count

Possible values

- A non-negative Integer

Property for

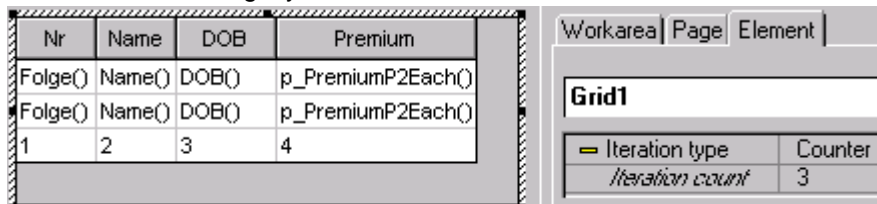
- Grid . Iteration type = Counter

Definition

The number of iterations in a Grid.

Example

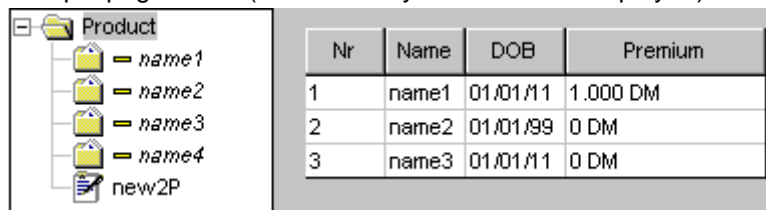
Assume the following layout



Nr	Name	DOB	Premium
Folge()	Name()	DOB()	p_PremiumP2Each()
Folge()	Name()	DOB()	p_PremiumP2Each()
1	2	3	4

Workarea Page Element	
Grid1	
- Iteration type	Counter
<i>/iteration count</i>	3

In the runtime layout, note that 3 iterations are always included in the Grid, regardless of how many multiple pages exist (note that only the first 3 are displayed).



Nr	Name	DOB	Premium
1	name1	01.01.11	1.000 DM
2	name2	01.01.99	0 DM
3	name3	01.01.11	0 DM

Property: Iteration property

Possible values

- Property.

Property for

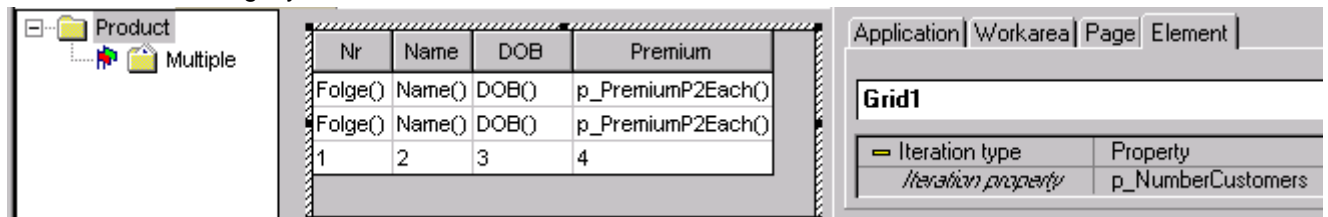
- Grid . Iteration type = Property

Definition

The property that controls iteration.

Example

Assume the following layout Grid:



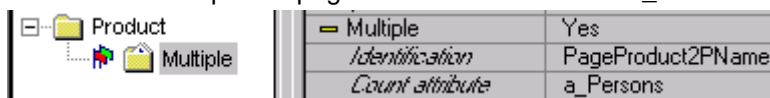
The screenshot shows a 'Product' folder containing a 'Multiple' subpage. A grid is displayed with the following data:

Nr	Name	DOB	Premium
Folge()	Name()	DOB()	p_PremiumP2Each()
Folge()	Name()	DOB()	p_PremiumP2Each()
1	2	3	4

Below the grid, the 'Grid1' properties are shown:

Property	Value
Iteration type	Property
Iteration property	p_NumberCustomers

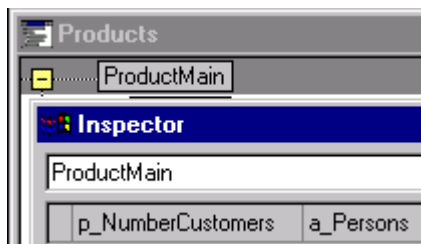
Assume the multiple subpages <Count attribute> = <a_Persons>.



The screenshot shows the 'Multiple' subpage properties:

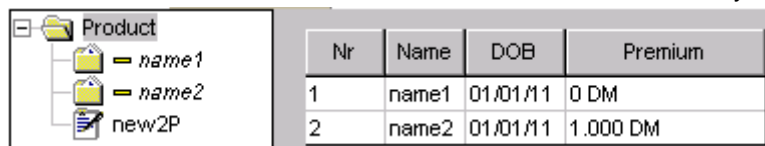
Multiple	Yes
Identification	PageProduct2PName
Count attribute	a_Persons

Assume the model



The screenshot shows a 'Products' model structure with 'ProductMain' and an 'Inspector' window showing 'ProductMain' with properties 'p_NumberCustomers' and 'a_Persons'.

In the runtime model, the number of iterations is determined by the value of p_NumberCustomers.



The screenshot shows a 'Product' folder with subpages 'name1', 'name2', and 'new2P'. A grid is displayed with the following data:

Nr	Name	DOB	Premium
1	name1	01.01.11	0 DM
2	name2	01.01.11	1.000 DM

Property: Iteration type

Possible values

- Attribute
- Property
- Counter

Property for

- Grid

Definition

The iteration type.

Example

For the following Grid, the number of iterations would be dynamic and = a_Persons

Iteration type	Attribute
<i>Attribute</i>	a_Persons

For the following Grid, the number of iterations would be dynamic and = model product property p_NumberCustomers:

Iteration type	Property
<i>Iteration property</i>	p_NumberCustomers

For the following Grid, the number of iterations would always = 3:

Iteration type	Counter
<i>Iteration count</i>	3

Property: Key column

Possible values

- Positive Integer

Property for

- Graphics
- GridList

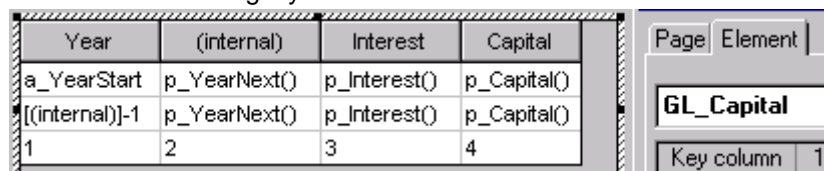
Definition

For Grid: Specifies the Grid key column. The key column is the index for the data array.

For Graphics: Specifies the source Grid key column for the Graphics x-axis pointer.

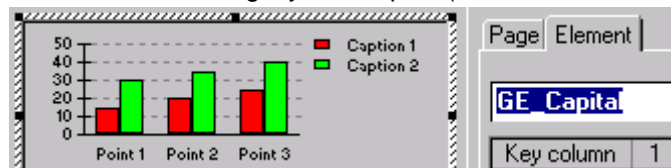
Example

Assume the following layout Grid:



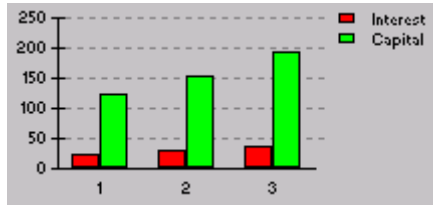
Year	(internal)	Interest	Capital
a_YearStart	p_YearNext()	p_Interest()	p_Capital()
[(internal)]-1	p_YearNext()	p_Interest()	p_Capital()
1	2	3	4

Assume the following layout Graphic (with the above Grid as the source):



In the runtime layout:

Year	(internal)	Interest	Capital
1	2	25 DM	125 DM
2	3	31 DM	156 DM
3	4	39 DM	195 DM



Property: Left

Possible values

- Non-negative Integer

Property for

- PushButton . Action = Dialog-Call

Definition

Number of pixels that the subdialog left edge is to the right of the calling dialog (with the PushButton) left edge.

Example

Assume the following layout:



In the runtime layout, the subdialog upper left corner would correspond with the calling dialog upper left corner:



With <Left> = 50:



Property: Legend placement

Possible values

- Right
- Bottom

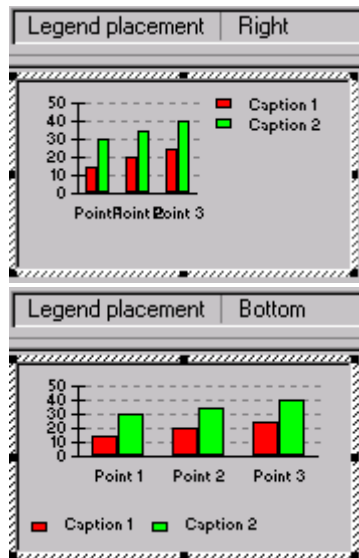
Property for

- Graphics

Definition

Location of the graphic legend.

Examples



Property: Level

Possible values

- Level0 ... Level7

Property for

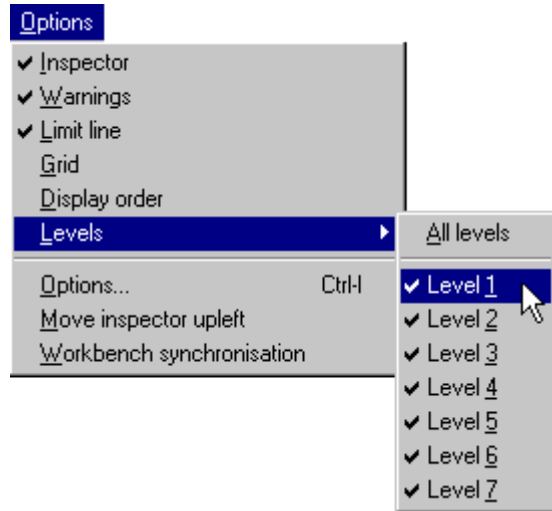
- 3DBorder
- Border
- CheckBox
- ComboBox
- EditField
- ExtendedGridList
- Graphics
- GridList
- Label
- Line
- Multimedia
- PushButton
- RadioButton
- RadioGroup
- ResultField

Definition

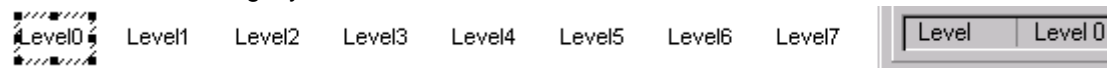
The level on which the element is located.

Example

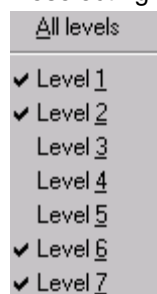
Assume that all levels are displayed:



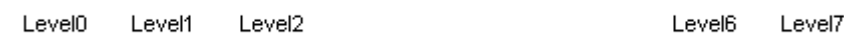
Assume the following layout:



Deselecting levels 3-5



would result in the following:



Property: List (...;...)

Possible values

- A list of model product results separated by semicolons.

Property for

- EditField . ChkOwnAvailability = List
- ComboBox . ChkOwnAvailability = List
- CheckBox . ChkOwnAvailability = List
- RadioButton . ChkOwnAvailability = List
- PushButton . ChkOwnAvailability = List
- Label . ChkOwnAvailability = List
- Border . ChkOwnAvailability = List
- RadioGroup . ChkOwnAvailability = List

Definition

Specifies a result list for availability.

Property: Mask

Possible values

Any combination of the following symbols:

- # (digit)
- N (digit; default = 0)
- & (digit or system decimal limiter)
- X (letter)
- C (letter; default = [space])
- U (letter; converted to upper case on entry)
- L (letter; converted to lower case on entry)
- ? (any character)
- / (signifies that the next character is a delimiter)

Property for

- EditField . Type = Mask
- Extended Grid . Property for = Column (column number) . Usage = Input . Type = Mask
- ResultField . Type = Mask result

Definition

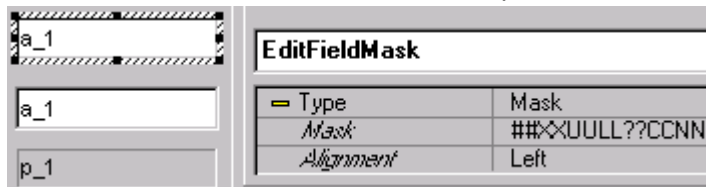
For EditField: Defines the acceptable input and how that input maybe be modified upon entry.

For ResultField: Defines how the result will be modified before output.

Example

Assume the following layout:

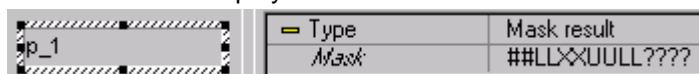
The first EditField uses a mask to mask input:



The second EditField displays (without a mask) the attribute entered in the first mask.



The ResultField displays with a mask the attribute entered in the first mask.



Upon opening the runtime layout:



Placing the cursor at the start of the top EditField and then pressing any combination of character keys will have no effect (the first 2 digits must be numbers (##)).

After the first 2 number have been entered, then any 2 letters must be entered (XX).

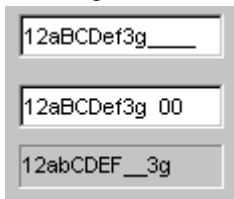
Next any 2 letters must be entered; the letters will be converted to uppercase (UU).

Next any 2 letters must be entered; the letters will be converted to lowercase (LL).

Next any 2 numbers/letters can be entered (??).

The last 4 characters do not have to be entered; If not entered, they will automatically be set to 2 space characters and 2 zeros (CCNN).

Entering "12aBcDeF3g" would result in the following:



Note in the second Edit Field that a default of " 00" (space + space + 00) was added.

Note in the ResultField that "b" is lowercase, "E" is uppercase, and that an error was generated (extra space inserted) when 3 appeared in a location that should be displayed in lowercase.

Property: MaxFrameWidth

Possible values

- Integer.

Property for

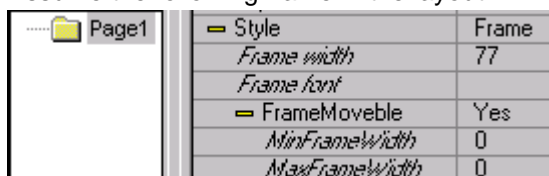
- Workarea . Style = Frame . FrameMoveable = Yes

Definition

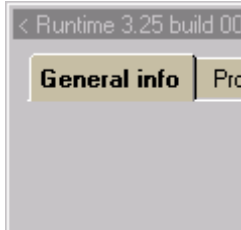
Specifies the maximum frame width.

Example

Assume the following frame in the layout:



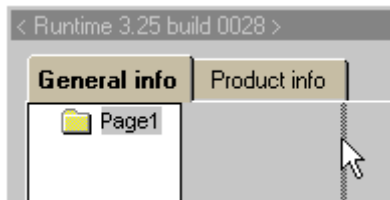
In the runtime, the frame width can be changed to any size. In the following example, the right edge of the frame is to the left of the left edge of the workarea (the frame width is actually negative):



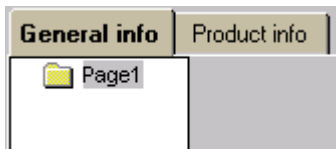
Assume the following frame in the layout:

Page1	Style	Frame
	Frame width	77
	Frame font	
	FrameMoveable	Yes
	MinFrameWidth	30
	MaxFrameWidth	90

In the runtime layout, the right edge of the frame can be dragged any distance to the right:



However, after releasing the right mouse button, the right edge of the frame will be set at <MaxFrameWidth>.



Property: Maximum

Possible values

- Non-negative Integer

Property for

- EditField . Type = Currency / Number / Text
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Currency / Number / Text

Definition

The maximum number of digits/characters that can be entered.

Note: If <Maximum> = 0: No maximum limit is set.

Example

Assume the following layout with a text EditField with <Maximum> = 10.



In the runtime layout, a maximum of 10 characters can be entered:

<Maximum> = 10

1234567890

Property: Media file

Possible values

- Path and name of media file.

Property for

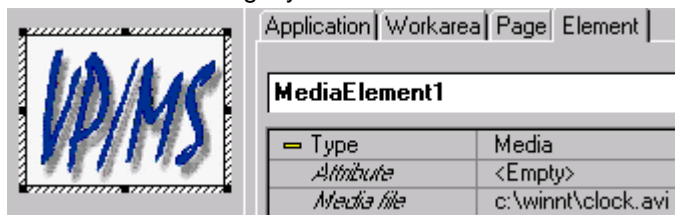
- Multimedia . Type = Media

Definition

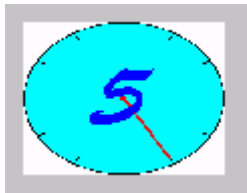
Specifies the media file.

Example

Assume the following layout:



In the runtime layout the .avi file would be played:



Property: MinFrameWidth

Possible values

- Integer.

Property for

- Workarea . Style = Frame . FrameMoveable = Yes

Definition

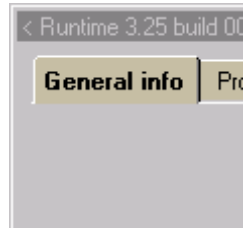
Specifies the minimum frame width.

Example

Assume the following frame in the layout:

Page1	Style	Frame
	Frame width	77
	Frame font	
	FrameMoveable	Yes
	MinFrameWidth	0
	MaxFrameWidth	0

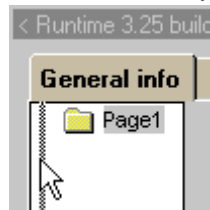
In the runtime, the frame width can be changed to any size. In the following example, the right edge of the frame is to the left of the left edge of the workarea (the frame width is actually negative):



Assume the following frame in the layout:

Page1	Style	Frame
	Frame width	77
	Frame font	
	FrameMoveable	Yes
	MinFrameWidth	30
	MaxFrameWidth	90

In the runtime layout, the right edge of the frame can be dragged any distance to the left:



However, after releasing the right mouse button, the right edge of the frame will be set at <MinFrameWidth>.



Property: Minimum

Possible values

- Non-negative Integer

Property for

- EditField . Type = Currency / Number / Text
- ExtendedGridList . Property for = Column 1 . Usage = Input . Type = Currency / Number / Text

Definition

The minimum number of digits/characters required for entry. If the minimum number is not entered: The focus cannot be switched to another layout component.

Note: If <Minimum> = 0: No minimum is set.

Example

Assume the following layout with a text EditField with <Minimum> = 5.



In the runtime layout: If less than 5 characters have been entered in the EditField: Clicking on another element does not change the focus.

Property: Multiline

Possible values

- Yes
- No

Property for

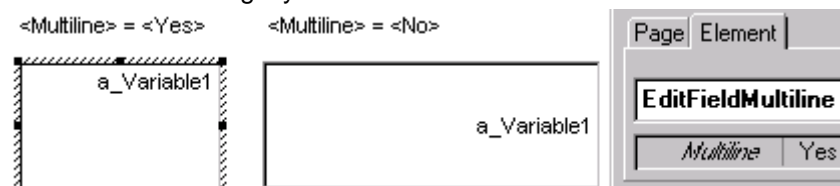
- CheckBox . Type = CheckBox
- RadioButton . Type = RadioButton
- EditField . Type = Text
- Label . Type = Label
- ResultField . Type = Result

Definition

If <Yes>: The element can accept multiline input (with each line separated by a carriage return).

Example

Assume the following layout:



Note that the left EditField is multiline and that the right EditField is single line.

In the runtime layout entering the following lines

- a very very long first line
-
- third line
- fourth line

would result in the following:

<Multiline> = <Yes>

<Multiline> = <No>

```

a very very long
  first line

  third line

```

```

a very very long first line

```

Note: To view the text "fourth line": Click in the left EditField and scroll down.

Property: Multiple

Possible values

- No
- Yes

Property for

- Page

Definition

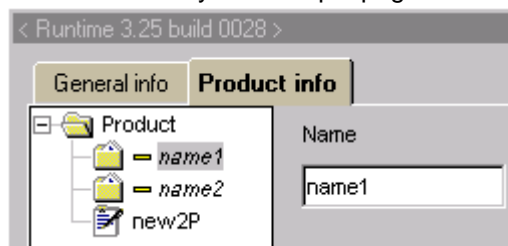
If <Yes>: Multiple pages can be created.

Example

Assume the following page with <Multiple> = <Yes>.



In the runtime layout: Multiple pages can be created:



Property: Name

Possible values

- Model attribute name

Property for

- Application . StartAttr = Constant / Dynamic

Definition

The name of the application start attribute (a model attribute).

Example

Assume the following layout:



When the runtime layout is opened, the model attribute `a_Variable2 = 1234`:



Property: Name (Workarea/Page)

Possible values

- Model attribute name

Property for

- Workarea . Visible = Property
- Page . Visible = Property

Definition

If the model attribute specified by `<Name>` is an Integer `> 0`: The Workarea / Page is visible.

Property: New Session

Possible values

- No
- Yes

Property for

- PushButton . Action = Dialog-Call

Definition

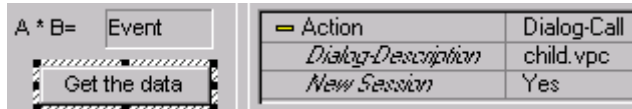
If `<Yes>`: The dialog opened by clicking on the PushButton is opened in a different session.

This property indicates, whether the secondary dialog uses the same VP/MS session as the parent application (value No), or opens a new session using the VP/MS model name described in its VPL-file

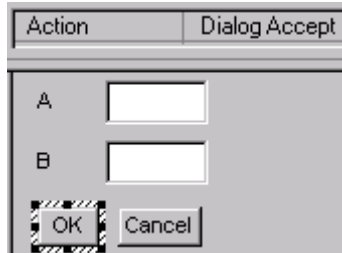
(value Yes). If the secondary dialog uses a new session, it will have no any impact on the state of the VP/MS model of the parent application. Default value is No.

Example

Assume the following "parent" layout with the PushButton property <New Session> = <Yes>:



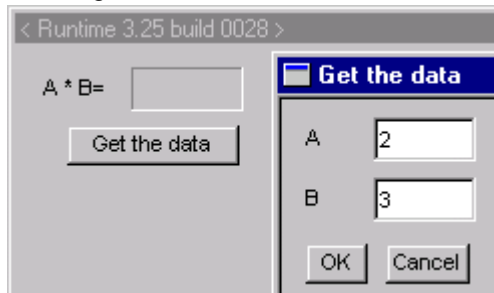
Assume the following "child" layout:



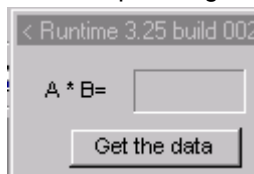
Assume the following model:



In the runtime layout, clicking on the "Get the data" PushButton opens a child dialog in a new session. Entering the values for A and B

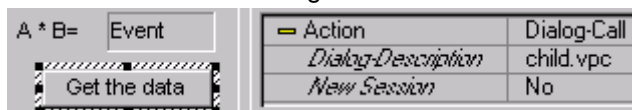


and then pressing the "OK"

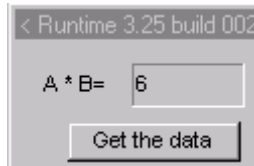


Note that the data was entered in a new session, and therefore is not available in the parent dialog.

If <New Session> is changed to <No>



and the above steps are repeated, then the result is calculated:



Property: Next line

Possible values

- Positive Integer.

Property for

- Grid . Property for = Column (column number) . Usage = Input

Definition

Specifies the column that uses the contents of the selected column for generating the index value for an array.

Example

Assume the following Grid in the layout:

Nr	Name	DOB	Premium	Folge
Folge()	Name()	DOB()	p_PremiumP2Each()	Start
Folge()	Name()	DOB()	p_PremiumP2Each()	[Nr]-1
1	2	3	4	5

Property for	Column 5
Title	Folge
Width	50
Usage	Input
Attribute	Start
Next line	1

For column 5 <Next line> = 1. Thus the value in column 5 will be used as the input into the model equation that defines the value for column 1 (Folge()).

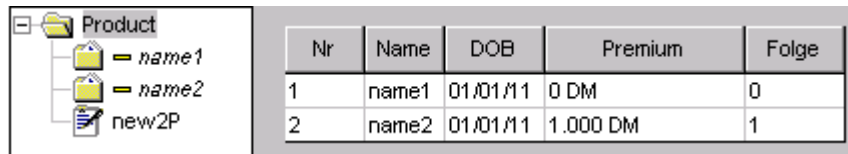
The next value for column 5 comes from the previous row in the column that generated the value (column 1). Therefore the value in row 2 for column 5 is "[Nr]-1".

<Next line> = 1. Therefore, the value for column "Folge" in the second row will be the value from column "Nr" in the previous row ("[Nr]-1").

Assume the following model with Folge(i) = i+1 (i = iteration counter).



In the runtime layout, the value for row 1 of column 1 is 1 (0 from row 1 column 5 plus 1). The value for row 2 of column 5 is from row 1 of column 1 (1). Thus the value generated for row 2 of column 2 is 2 (1 + 1).



Property: Number of columns

Possible values

- Positive Integer.

Property for

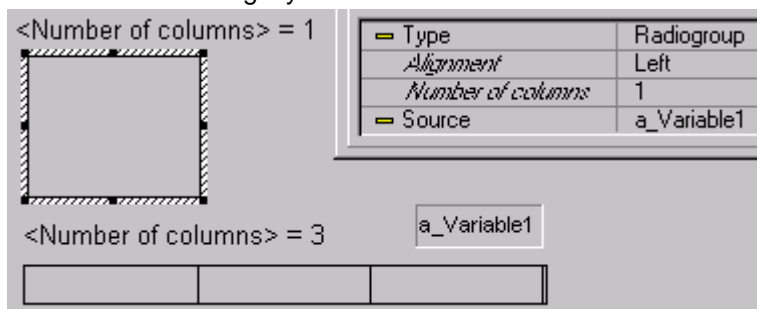
- RadioGroup . Type = RadioButton

Definition

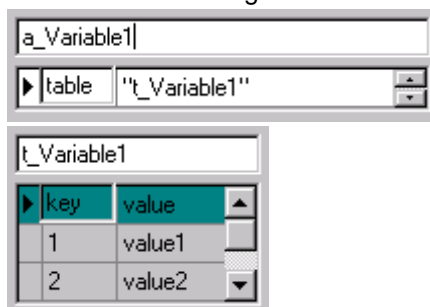
Number of horizontal colums for the RadioGroup.

Example

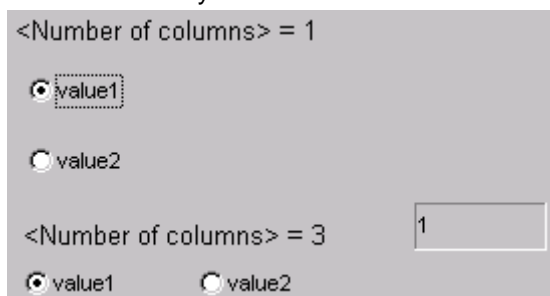
Assume the following layout:



Assume the following model:



In the runtime layout:



Property: Off conversion List

Possible values

- Name of a model attribute

Property for

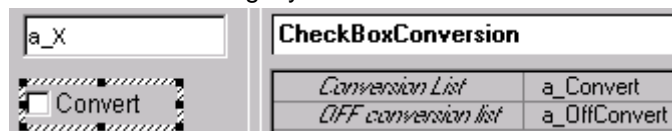
- CheckBox . Type = CheckBox

Definition

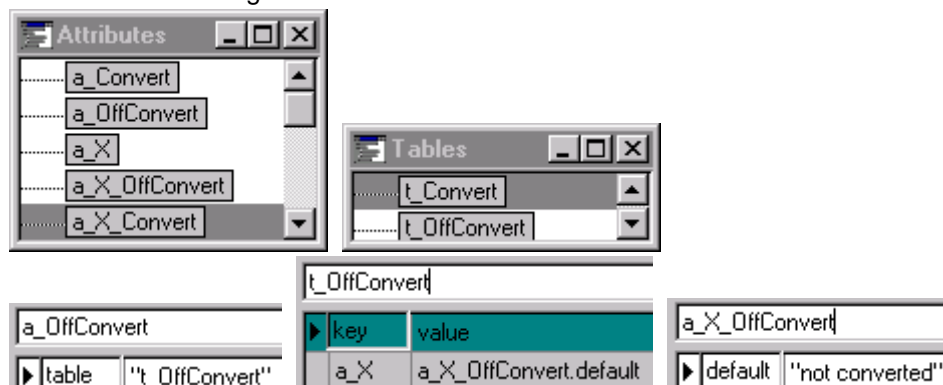
Specifies the model attribute whose property <table> references the (OFF) conversion table when the CheckBox is NOT selected.

Example

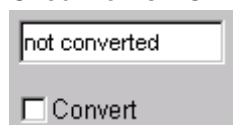
Assume the following layout:



Assume the following model:



In the runtime model, the attribute a_X would be converted to a_X_OffConvert.default when the CheckBox is NOT checked:



Property: Off symbol

Possible values

- filename (of symbol .bmp file)

Property for

- Page . DataIndicator = On

- Workarea . DataIndicator = On

Definition

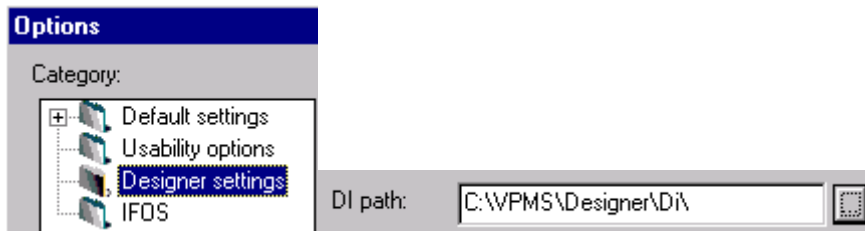
The symbol .bmp file contains 2 icons for indicating when all required data has been entered and when not entered.

See also:

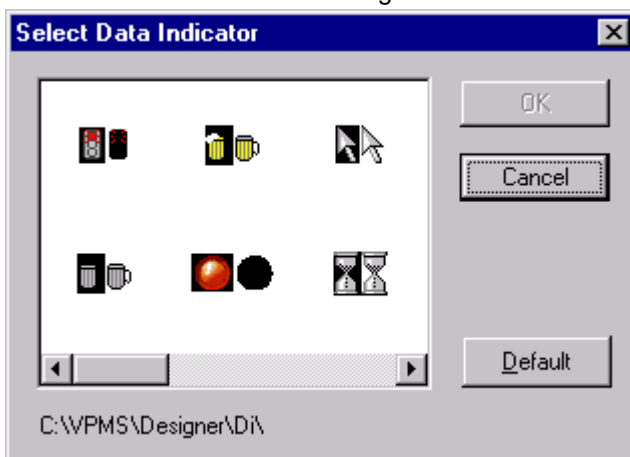
- Specifying directory of symbol files (DI path).

Example

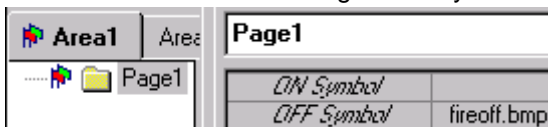
Assume the following DI path:



Clicking in the <Off symbol> property area in the Inspector and clicking on the resulting button opens the "Select Data Indicator" dialog:

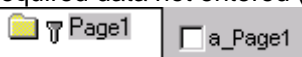


Clicking on a DI icon and clicking on "OK" selects the DI icon for the element. Note, however, that the icon is not shown in the design-time layout:

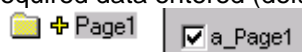


In the runtime layout:

- Required data not entered (fireoff.bmp icon):



- Required data entered (default icon):



Property: On symbol

Possible values

- filename (of symbol .bmp file)

Property for

- Page . DataIndicator = On
- Workarea . DataIndicator = On

Definition

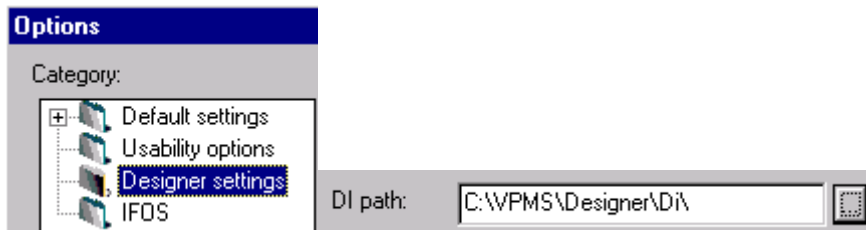
The symbol .bmp file contains 2 icons for indicating when all required data has been entered and when not entered.

See also:

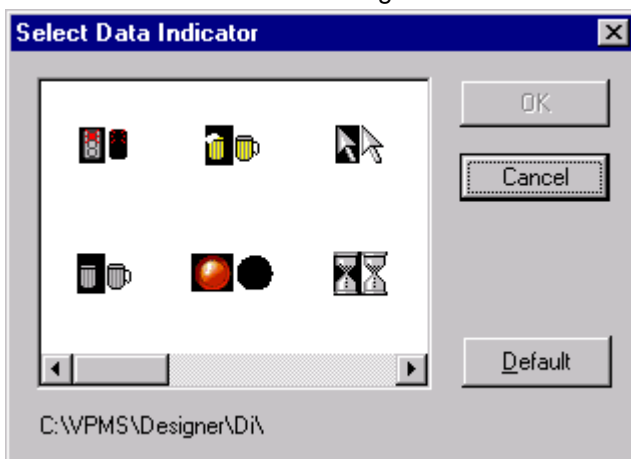
- Specifying directory of symbol files (DI path).

Example

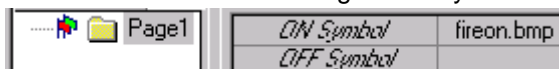
Assume the following DI path:



Clicking in the <On symbol> property area in the Inspector and clicking on the resulting button opens the "Select Data Indicator" dialog:

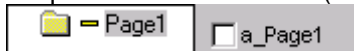


Clicking on a DI icon and clicking on "OK" selects the DI icon for the element. Note, however, that the icon is not shown in the design-time layout:



In the runtime layout:

- Required data not entered (default icon):



- Required data entered (fireon.bmp icon):



Property: Order

Possible values

- Individual
- Standard

Property for

- Page

Definition

The order in which the pages are displayed.

Property: Position (Footer)

Possible values

- Right
- Bottom

Property for

- Workarea

Definition

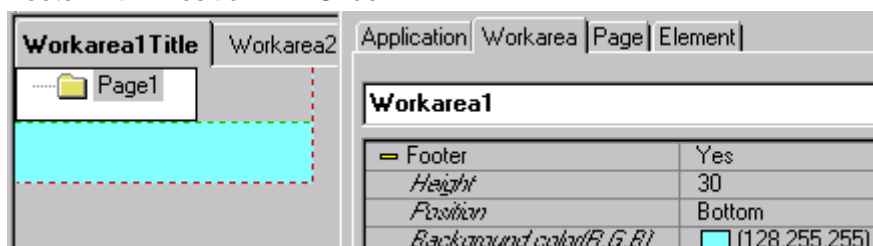
Specifies the position of the Footer in relation to the workarea.

Example

Footer with <Position> = <Right>.



Footer with <Position> = <Under>.



Property: Position (Header)

Possible values

- Right
- Top

Property for

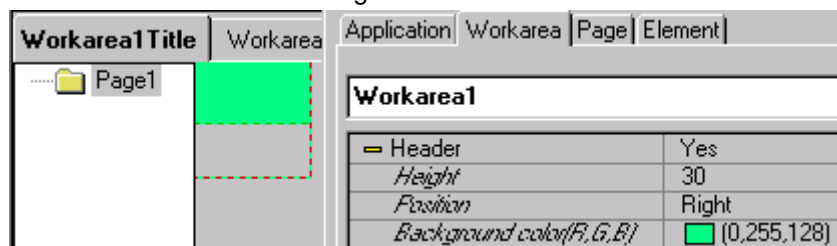
- Workarea

Definition

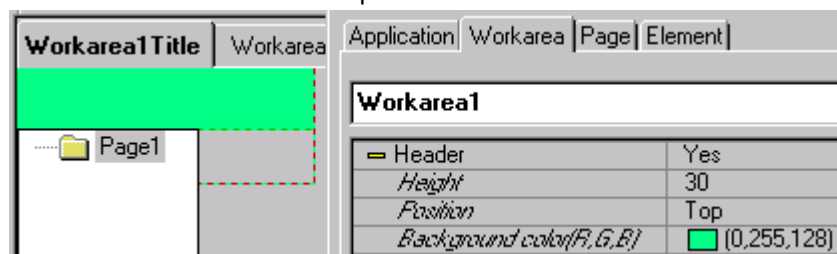
Specifies the position of the header in relation to the workarea.

Example

Header with <Position> = <Right>.



Header with <Position> = <Top>.



Property: Productmodel

Possible values

- Path and name of .vpm file.

Property for

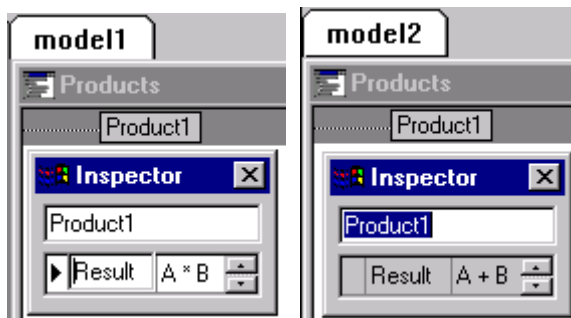
- Application

Definition

The compiled product model (.vpm) linked to the layout.

Example

Assume the 2 models:



Assume the following layout with `<Productmodel> = <model1.vpm>`:



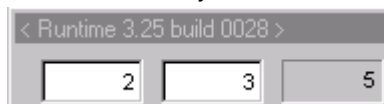
In the runtime layout:



Changing to `<Productmodel> = <model2.vpm>`:



In the runtime layout:



Property: Property

Possible values

- A model property to links to a model attribute.

Property for

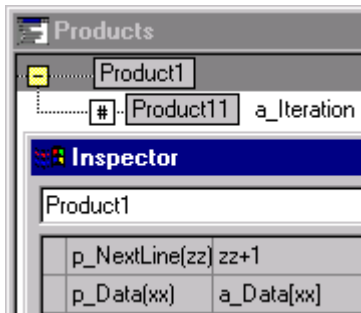
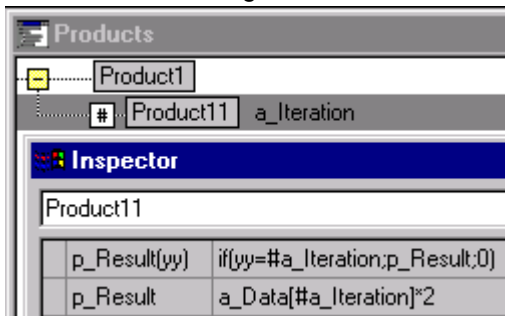
- ExtendedGrid . Visible = Value
- Grid . Property for = Column 1 . Usage = Output
- Grid . Visible = Value
- ResultField . Source = <Empty> . Visible = Value

Definition

The model property that provides access to the model variable.

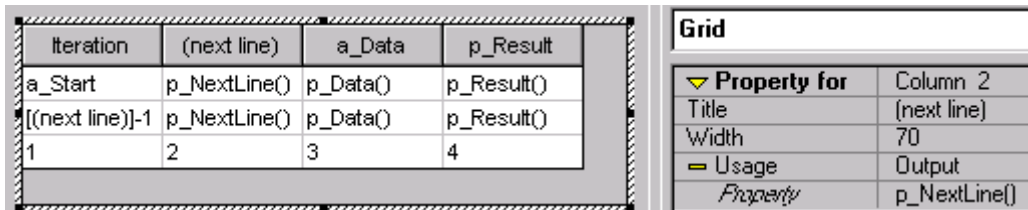
Example

Assume the following model:



Note the model properties provide access to the model attributes.

Assume the following layout Grid.



Note that column 2 displays p_NextLine().

Column 3 displays p_Data() (<Property>=<p_Data()>).

Column 4 displays p_Result() (<Property>=<p_Result()>).

Property: Property for

Possible values

- Column.

Property for

- Grid
- ExtendedGrid
- Graphics

Definition

Specifies the column that the properties are listed for.

Example

Assume the following layout ExtendedGrid:

Nr	Name	Geburtsdatum	Premium
Folge()		a_DOB	p_PremiumP2Each()
1		3	4

The following shows the properties for column 1 only. Each column has its own set of these properties.

Property for	Column 1
Title	Nr
Width	40
Usage	Output
Type	Text
Alignment	Left
Property	Folge()
Indexed	Yes
Visible	Yes

Property: Series

Possible values

- Integer.

Property for

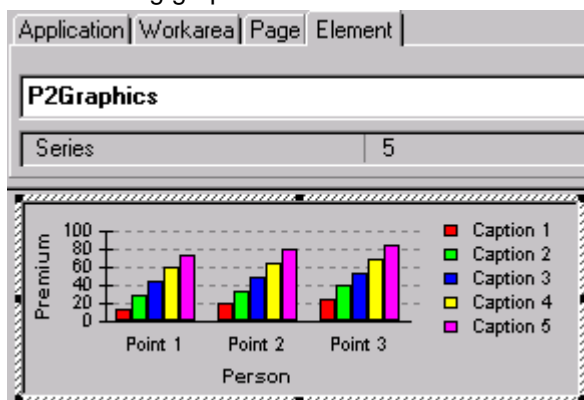
- Graphics

Definition

The number of series in the graphic.

Example

The following graphic has 5 series.



Property: Source

Possible values

- Attribute.

Source

- CheckBox
- ComboBox
- EditField
- PushButton
- RadioButton
- RadioGroup
- ResultField

Definition

The source attribute for the element.

Example

The following EditField has model property a_Variable1 as its source.



Property: StartAttr

Possible values

- No
- Constant
- Dynamic

Property for

- Application

Definition

The type of start attribute for the application.

Example

Following layout with a constant StartAttr:



When the runtime layout is open, model attribute a_Variable2 = 1234:



Property: Strict check

Possible values

- No
- Yes

Property for

- EditField . Type = Currency / Date / Mask / Number / Text
- ExtendedGrid . Property for = Column 1 . Usage = Input . Type = Currency / Date / Mask / Number / Text

Definition

Specifies if a strict check.

Property: Style (application)

Possible values

- Notebook
- Simple

Property for

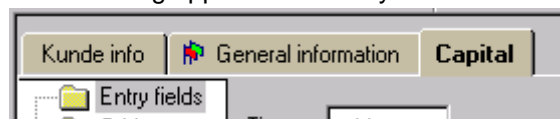
- Application

Definition

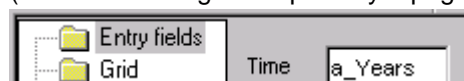
Specifies the application style (notebook or simple).
Default setting.

Example

The following application has style Notebook:



The following application has style Simple. Note that the other notebook pages were simply deleted (when converting to simple only 1 page can remain).



Property: Style (workarea)

Possible values

- Simple
- Frame

Property for

- Workarea

Definition

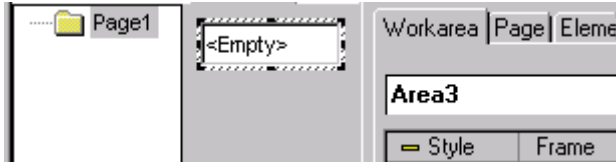
Specifies the workarea style (simple or frame)

Example

The following workarea has style Einfach:



The following workarea has style Frame. Note that when first created the frame contains only 1 page.



When changing from Style frame to Einfach, a prompt appears requesting which page will not be deleted (all other pages will be deleted).

Property: Tab font

Possible values

- font, font size

Property for

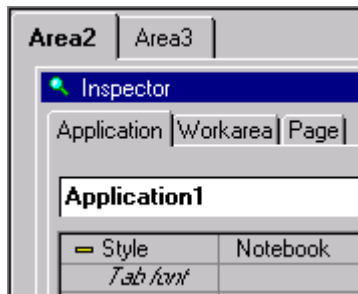
- Application . Style = Notebook

Definition

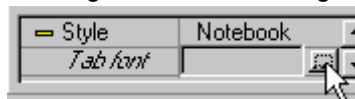
The font style for the workarea tabs.

Example

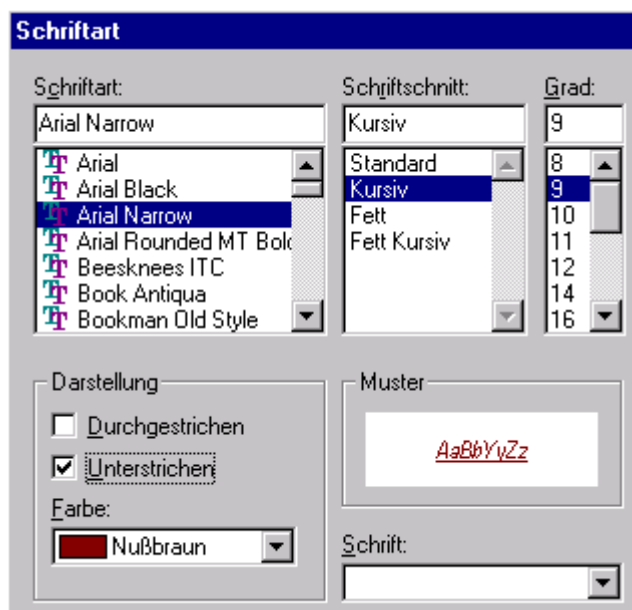
In the following layout the font for the workarea tabs is the default.



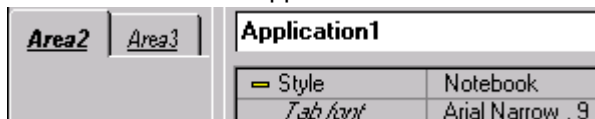
Clicking in the cell to the right of "Tab Schrift" causes a button to appear:



Clicking the button causes the "Schriftart" window to appear. The properties for the new schrift can be selected:



The new schrift then appears in the workarea tabs:



Note: To change the font back to the default value, reopen the Schriftart dialog, delete all settings, and click OK:

Property: Tab color

Possible values

- (0..255,0..255,0..255)

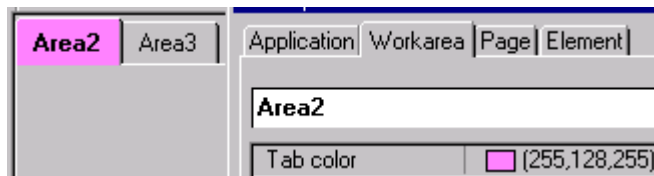
Property for

- Workarea

Definition

The color of the workarea tab.

Example



Property: Title

Possible values

- Text.

Property for

- Application
- Border . Type = Border
- CheckBox . Type = CheckBox
- GridList . Property for = Column 1
- ExtendedGridList . Property for = Column 1
- Label . Type = Label
- Page
- PushButton
- RadioButton . Type = RadioButton
- Workarea

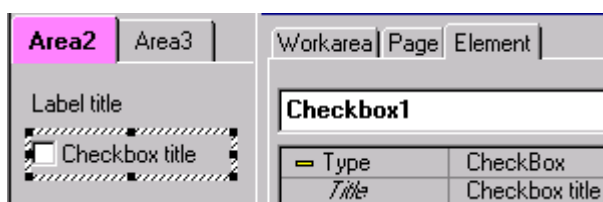
Definition

The title.

Example

The following layout contains:

- Workarea with <Title>=<Area2>.
- Workarea with <Title>=<Area3>.
- Label with <Title>=<Label title>.
- Checkbox with <Title>=<Checkbox title>.



Property: Tooltip (static/dynamic)

Possible values

- No (default)
- Static
- Dynamic

Property for

- CheckBox
- ComboBox
- EditField
- Label
- Multimedia
- PushButton
- RadioButton
- RadioGroup

Definition

Specifies a static tooltip or dynamic tooltip.

Property: Tooltip (static only)

Possible values

- No
- Yes

Property for

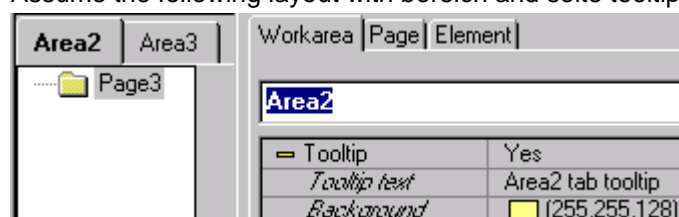
- ExtendedGridList
- Graphics
- GridList
- Page (if Bereich <Stil>=<Frame>)
- ResultField
- Workarea (if Anwendung <Stil>=<Notebook>)

Definition

Specifies a static tooltip.

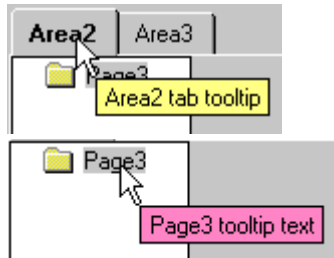
Example

Assume the following layout with bereich and seite tooltips.



Workarea Page Element	
Page3	
Tooltip	Yes
Tooltip text	Page3 tooltip text
Background	[255,128,192]

In the runtime layout:



Property: Tooltip text

Possible values

- Text (can include "/p" for multiline tooltips).

Property for

- CheckBox . Tooltip = Dynamic / Static
- ComboBox . Tooltip = Dynamic / Static
- EditField . Tooltip = Dynamic / Static
- ExtendedGridList . Tooltip = Yes
- Graphics . Tooltip = Yes
- GridList . Tooltip = Yes
- Label . Tooltip = Dynamic / Static
- Multimedia . Tooltip = Dynamic / Static
- Page . Tooltip = Yes
- PushButton . Tooltip = Dynamic / Static
- RadioButton . Tooltip = Dynamic / Static
- RadioGroup . Tooltip = Dynamic / Static
- ResultField . Tooltip = Yes
- Workarea . Tooltip = Yes

Definition

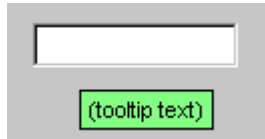
The tooltip text.

Use "/p" for a carriage return for multiple-line tooltips.

The tooltip properties Background color and font can also be specified.



Example

The following EntryField has property <Tooltip text> set to "(tooltip text)".

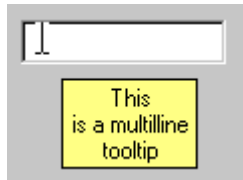


The following Eingabefeld has multi-line tooltip text.

Layout:

	<ul style="list-style-type: none"> Tooltip 	Static
	<i>Tooltip text</i>	This/ps a multiline/ptooltip
	<i>Background</i>	 (255,255,128)

Runtime layout:



Property: Top

Possible values

- Non-negative Integer

Property for

- `PushButton . Action = Dialog-Call`

Definition

Number of pixels that the subdialog top edge is below the calling dialog (with the PushButton) top edge.

Example

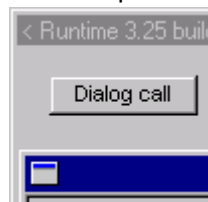
Assume the following layout:

	<i>Left</i>	0
	<i>Top</i>	0

In the runtime layout, the subdialog upper left corner would correspond with the calling dialog upper left corner:



With `<Top> = 50`:



Property: Type (entry elements)

Possible values

- Text
- Date
- Currency
- Number
- ComboBox
- RadioButton
- CheckBox
- RadioGroup
- Mask

Property for

The following list shows: Type of element (Corresponding property).

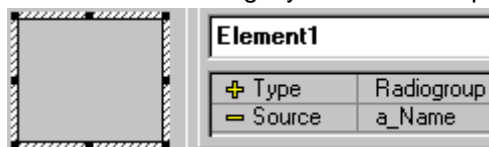
- Text EditField(Text)
- Date EditField(Date)
- Currency EditField(Currency)
- Number EditField(Number)
- ComboBox(ComboBox)
- RadioGroup(RadioButton)
- CheckBox(CheckBox)
- RadioGroup(RadioGroup)
- Mask (Mask)

Definition

Specifies the type of entry element. Changing the property will change the type of element.

Example

Assume the following layout RadioGroup

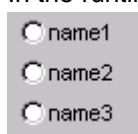


Assume the following model:

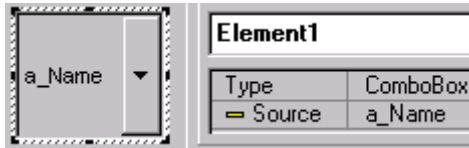
t_Name	
key	value
1	name1
2	name2
3	name3

Model structure: a_Name | table "t_Name"

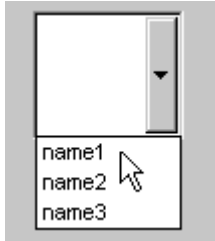
In the runtime layout:



Assume that the layout element <Type> is changed to <ComboBox>:



In the runtime layout:



Property: Type (result elements)

Possible values

- (Text) Result
- Number result
- Currency Result
- Date Result
- Mask (Result)

Property for

The following list shows: Type of element (Corresponding property).

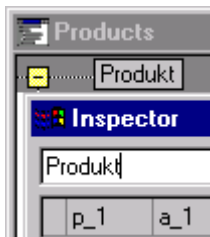
- (Text)Result (Result)
- NumberResult (Number result)
- CurrencyResult (Currency result)
- DateResult (Date result)
- MaskResult (Mask result)

Definition

Specifies the type of result element. Changing the property will change the type of element.

Example

Assume the following layout:



Assume the following layout:

Text	a_1
Result	p_1
Number result	111
Currency result	111
Date result	27.05.97

In the runtime layout:

Text	1234
Result	1234
Number result	1.234
Currency result	1.234 DM
Date result	

Text	1.2.1999
Result	1.2.1999
Number result	1
Currency result	1 DM
Date result	01.02.99

Property: Type (visual elements)

Possible values

- 3D-Area
- Border
- Label
- Line
- Media

Property for

- Label
- Border
- 3D Border
- Line
- Multimedia

Definition

Specifies the type of visual element. Changing this property will change the element accordingly.

Example

Assume the following layout Label:



Changing <Type> to <Border> changes the element to a Border with the same text for the title.



Property: Type (GraphicsElement)

Possible values

- "Graphics"

Property for

- GraphicsElement

Definition

Specifies the type of element.

Property: Type (Grid)

Possible values

- "Grid"

Property for

- Grid

Definition

Specifies the type of element.

Property: Type (ExtendedGrid)

Possible values

- "Extended Grid"

Property for

- ExtendedGrid

Definition

Specifies the type of element.

Property: Type (Grid column)

Possible values

- Text
- Number
- Currency

Property for

- Grid column

Definition

Specifies the type of data in the column.

Example

Assume the following layout Grid:

Nr	Name	DOB	Premium	Folge
Folge()	Name()	DOB()	p_PremiumP2Each()	Start
Folge()	Name()	DOB()	p_PremiumP2Each()	[Nr]-1
1	2	3	4	5

▼ Property for	Column 1
Title	Nr
Width	40
Usage	Output
Property	Folge()
Type	Number
Decimalposition	0
Alignment	Left

▼ Property for	Column 2
Title	Name
Width	40
Usage	Output
Property	Name()
Type	Text
Alignment	Left

▼ Property for	Column 3
Title	DOB
Width	50
Usage	Output
Property	DOB()
Type	Date
Format	dd/mm/yy
Alignment	Left

▼ Property for	Column 4
Title	Premium
Width	100
▣ Usage	Output
<i>Property</i>	p_PremiumP2Each()
▣ Type	Currency
<i>Currency-Symbol</i>	0n
<i>Decimalposition</i>	0
<i>Alignment</i>	Left

In the runtime layout:

Nr	Name	DOB	Premium	Folge
1	1	11.11.11	1.000 DM	0
2	2	11.11.11	0 DM	1
3			DM	2

Property: Type (ExtendedGrid column)

Possible values

- Text
- Number
- Currency
- Date
- Mask
- ComboBox (available if <Usage> = <Input>)

Property for


- ExtendedGrid column

Definition

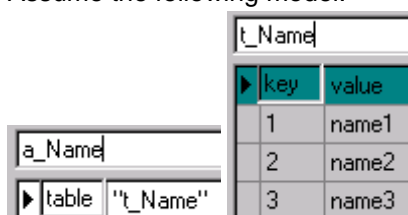
Specifies the type of EntryField/data in the column (the ComboBox contains text).

Example

Assume the following layout ExtendedGrid:



Assume the following model:



In the runtime layout, data can be entered for column 2 with a ComboBox:

Nr	Name	Date
1		
2	<div style="border: 1px solid black; padding: 2px;"> name1 name2 name3 </div>	

Property: Usage

Possible values

- Input
- Output

Property for

- ExtendedGridList . Property for = Column 1
- GridList . Property for = Column 1

Definition

The usage (input or output) of the column.

Example

Assume the following layout ExtendedGrid:

Nr	Name	Date of birth	Premium
Folge()	a_Name	a_DOB	p_PremiumP2Each()
1	2	3	4

▼ Property for	Column 2
Title	Name
Width	70
Usage	Input

▼ Property for	Column 4
Title	Premium
Width	100
Usage	Output

In the runtime layout: Data can be entered in column 2 and not entered in column 4.

Property: Value

Possible values

- Any alpha-numeric value

Property for

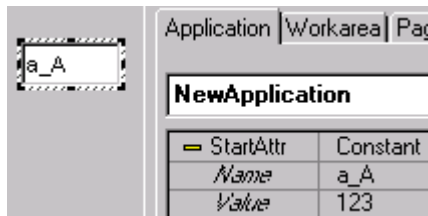
- Application . StartAttr = Constant

Definition

Specifies the value for the constant start attribute.

Example

Assume the following application with `<StartAttr> = <Constant>`, `<Name> = <a_A>`, and `<Value> = 123`.



When the runtime layout is opened:



Note: If model attribute `<a_A>` has property `<default>` defined: The value specified for the model default will be used.

Property: Version

Possible values

- Any text.

Property for

- Application . Version info = Yes
- Workarea . Version info = Yes
- Page . Version info = Yes
- EditField . Version info = Yes
- ComboBox . Version info = Yes
- CheckBox . Version info = Yes
- RadioButton . Version info = Yes
- PushButton . Version info = Yes
- Label . Version info = Yes
- Border . Version info = Yes
- Grid . Version info = Yes
- ExtendedGrid . Version info = Yes
- 3D Border . Version info = Yes
- Line . Version info = Yes
- Multimedia . Version info = Yes
- ResultField . Version info = Yes
- RadioGroup . Version info = Yes
- Graphic . Version info = Yes

Definition

The text entered as the version for the layout component in the dialog "Version information" (the version text cannot be entered in the Inspector field).

Example

Assume that the element version support default = <Yes>. If an EditField is added to the layout, <Version> = <1.0> by default.



This value can be changed in the "Version information" dialog. To open the dialog, click in the field that contains the version text and click on the resulting directory button.



Property: Version info

Possible values

- No
- Yes

Property for

- Application
- Workarea
- Page
- EditField
- ComboBox
- CheckBox
- RadioButton
- PushButton
- Label
- Border
- Grid
- ExtendedGrid
- 3D Border
- Line
- Multimedia
- ResultField
- RadioGroup
- Graphic

Definition

- If Yes: Commentary (version info) can be entered for the layout component.
- If set to No: Commentary for the layout component is deleted.

Example

For an example: See Commentary .

Property: Visible (element)

Possible values

- Yes
- No
- Check
- Value

Property for

- ExtendedGridList
- GridList
- ResultField . Source = <Empty>

Definition

Specifies the visibility of the element.

Examples

For examples of visibility see:

- Setting the default value.
- Concepts: Visibility.

Property: Visible (Workarea/Page)

Possible values

- Yes
- No
- Property

Property for

- Workarea
- Page

Definition

Specifies the visibility of the workarea/page.

Examples

For examples of visibility see:

- Setting the default value.
- Concepts: Visibility.

Property: Visualize

Possible values

- Selected
- Not selected

Property for

- CheckBox . Type = CheckBox
- RadioButton . Type = RadioButton

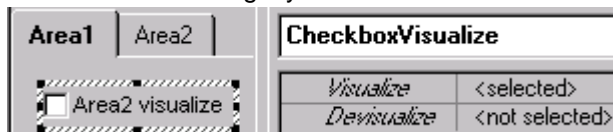
Definition

If <Selected>: A page / workarea has been selected as being "visualized" when the element is selected.

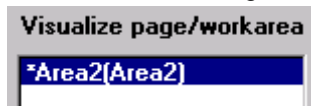
If <Not selected>: Selecting the element has no effect.

Example

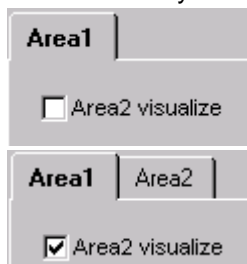
Assume the following layout:



Assume the following the selected pages/workareas for visualization:



The runtime layout:



Property: Width (application)

Possible values

- Non-negative Integer

Property for

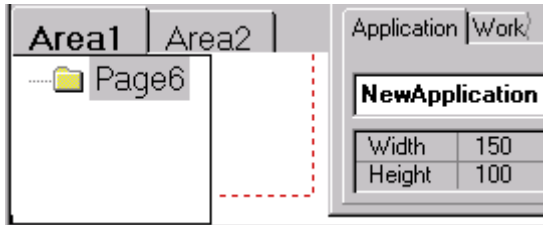
- Application

Definition

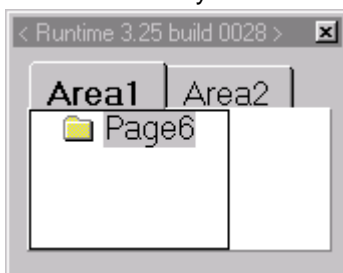
Specifies the width of the application.

Example

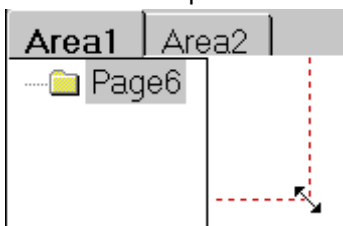
Assume the following layout:



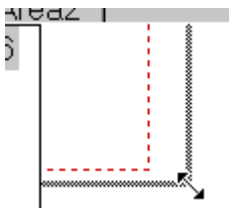
In the runtime layout



The mouse change be used to change the Width/Height. Move the mouse over the red-dotted line so that the mouse pointer becomes a double-arrow.



Press and hold the left mouse button while moving the mouse to change the application width/height.



Property: Width (Grid/ExtendedGrid)

Possible values

- Integer.

Property for

- Application
- ExtendedGridList . Property for = Column 1
- GridList . Property for = Column 1

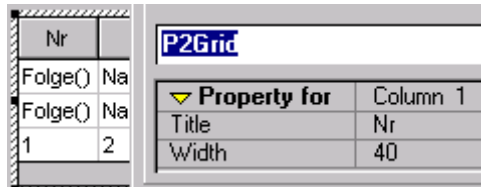
Definition

Width for a column.

Note: The width of the column can be changed in the runtime layout if element property <Columns moveable> = <Yes>.

Example

Assume the following layout Grid column:



Property: X-axis name

Possible values

- Text

Property for

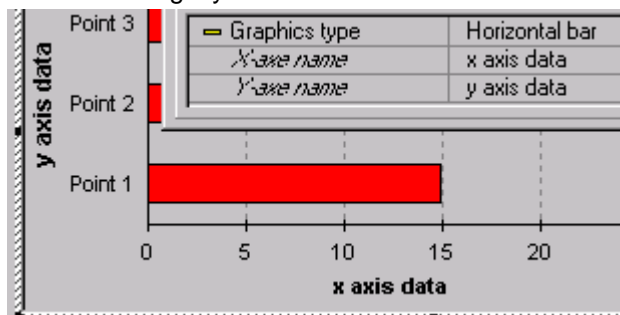
- Graphics . Graphics type = Horizontal 3D bar / Horizontal bar / Horizontal Gantt / Line / Vertical 3D bar / Vertical bar / Vertical Gantt

Definition

Name of the x axis in the graphic.

Example

In the following layout <X-axis name> = <x axis data>.



Property: Y-axis name

Possible values

- Text

Property for

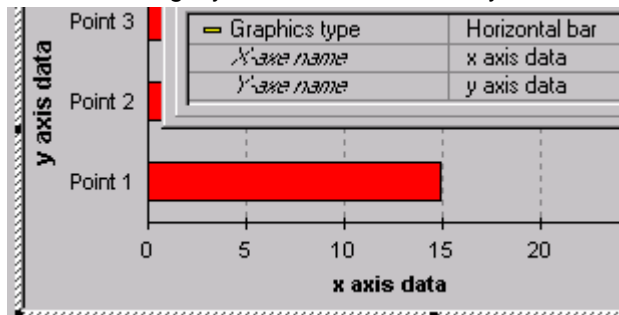
- Graphics . Graphics type = Horizontal 3D bar / Horizontal bar / Horizontal Gantt / Line / Vertical 3D bar / Vertical bar / Vertical Gantt

Definition

Name of the y axis in the graphic.

Example

In the following layout <Y-axis name> = <y axis data>.



Trouble-shooting

Click on the any problem listed below for a description of the most common solutions for that problem.

- The upper-left portion of a layout window is hidden in the Designer (window cannot be moved).
- An element is not visible in the design-time layout.
- A model attribute is not available from the drop-down list for an element.
- The header and/or footer for the application and/or workarea is not visible.
- The header and/or footer for the application and/or workarea is missing (the layout was previously saved an upon re-opening the header and footer were missing).

Window not visible

Problem

It is possible that a layout can be opened in the layout with the layout shifted to the upper-left such that the dialog cannot be moved with a mouse. For example:

Solution

Select from the main menu "Window / Cascade".

Element not visible

Problem

An element exists in the layout, but is not visible in the design-time (not runtime) layout. For example, the following layout has 2 Labels, but only Label1 is visible.

Solution

The non-visible element is probably on a level that is not displayed in the design-time layout. In the above example:

Select "Options / Levels / Level 2" to display the level. The element is now visible.

Model attribute not available

Problem

You want to select a model attribute as the <Source> for an element. However, the attribute is not in the drop-down list for the element property <Source>.

For example, you want to select model 333.pms attribute a_3 as the source for an EntryField in a layout. However, a_3 is not available from the drop-down list:

Requirement 1

The model must be selected for application property <Productmodel>.

Requirement 2

The model must be opened in the Workbench and be the currently selected model.

Example

Assume that <Productmodel> = 333.vpm.
Assume model 333.vpm includes 444.vpm:

Although 333.vpm is specified as <Productmodel>, only 444.vpm attribute a_4 will be available from the drop-down list if model 444 is selected in the Workbench. Model 333 must be selected (the selected model tab is white).

Requirement 3

The runtime model is selected as the <Productmodel>, not the model. Therefore if an attribute has been added to a model, the model be compiled to a runtime model.

Requirement 4

If an attribute has been added to the runtime model since the layout was opened or the last time the layout was synchronized with the model, then the layout must be synchronized with the model by selecting from the main menu "Options / Workbench synchronization".

Header/footer (application/workarea) not visible

Problem

The header and/or footer for the application and/or workarea is not visible. For example:

Solution

Click on the required Header or Footer icons in the Header/Footer toolbar.
The header and footer will be displayed.

Header/footer missing

Problem

You opened a layout and the header or footer for the application or workarea is missing.

Solution

A header or footer must contain at least 1 element to be saved to the .vpl file.

Example

Assume the following layout with an application header and footer:

Closing (and saving) the layout and then reopening would result in the following layout:

FAQ (frequently-asked questions)

The following is a list of frequently-asked questions.

- Can different elements be displayed in the same place in the runtime layout?
- Can attributes from more than 1 model be included in a single layout?

Overlapping elements

Question

How can the element in a runtime layout change?

Answer

The element in the runtime layout can change by specifying overlapping elements in the design-time layout.

Only 1 of the overlapping elements is visible at any time in the runtime layout.

The overlapping elements are edited in the design-time layout by using levels.

Example

Assume the following layout:

Assume the following model:

In the runtime layout, either the EntryField or the ComboBox would be visible:

The problem at this point is how to edit the 2 elements in the design-time layout if they are to be overlapping. This is solved by using levels.

Note the following settings for the EntryField:

The settings for the ComboBox are changed to the following:

Note that the levels for the 2 elements differ. The EntryField is normally shown on top of the CheckBox (level 1 on top of level 2).

To display the ComboBox (level 2), select (uncheck) from the main menu "Options / Levels / Level 1". All components on level 1 are hidden and the ComboBox is visible:

NOTE: Level 0 can NOT be hidden.

Attributes from more than 1 model

Question

How can elements from more than 1 model be included in the layout?

Answer

Only a single runtime model (.vpm) can be specified for the layout property <Productmodel>. However, the model specified for the layout can include other models. The attributes from the included models are also available in the layout.

Glossary

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 vehicle). 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 OEC catalog

The objects, events and compensation which can be used in products are each defined in a basic catalog (Objects window, Events window, Compensation 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 calculations 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 conditions 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 product modeler can inspect all entries in detail by double-clicking all elements 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.

Product

Insurance products can be represented in the form of a tree which contains the elements product, objects, events and compensation (in this order). This tree-like representation of an insurance product can be created 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 complete 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 = vpmstdll.dll), Windows 95 (32-bit runtime = vpmstdll32.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".

VP/MS

VP/MS means Virtual Product / Modeling System and is a tool for insurance 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 applications, 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.

Directories and files

To understand how to best utilize Designer, to find solutions to problems, etc., it is important to understand the following in Designer:

- Directories
- Files
- File types

Directories

The following is a list of directories created for VPMS / Designer on your computer.

Note: The directory name "ZYX" is associated with VPMS application ZYX. If you have created or installed other applications, then directories with the corresponding name will be created for these applications.

- c:\vpms
- c:\vpms\designer
- c:\vpms\vframe
- c:\vpms\vframe\menu
- c:\vpms\wbench
- c:\vpl_apps
- c:\vpl_apps\zyx
- c:\vpl_apps\zyx\data
- c:\vpl_apps\zyx\help
- c:\vpl_apps\zyx\images
- c:\vpl_apps\zyx\print
- c:\vpl_apps\zyx\struct
- c:\vpl_apps\zyx\vorgang

c:\vpms

The main VPMS application directory.

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.

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.

Files

The following is a list of files associated with Designer (and VPMS in general).

- cafrg.exe
- consult.ini
- empty.vpm
- ifosre.exe
- menubar.ini
- menuburo.ini
- menufens.ini
- menuhelp.ini
- menuinfo.ini
- menukund.ini
- vds.exe
- vds_tx32.exe
- verwalt.ini
- vframe32.exe
- vorgang.ini
- vpms32.exe
- vpmsdl32.dll
- vpmsdll.dll
- vpmsin32.dll
- vpmsinfo.dll
- vpmste32.exe
- xplconf.ini

cafrg.exe

Description

Report viewer exe.

Directory

C:\VPMS\Wbench\

consult.ini

Description

Definition of submenu items for VFrame main menu item Consultation.

Directory

C: \ VPMS \ Vframe \ menu \

empty.vpm

Description

The default runtime model for layouts. This model is used if layout property <Product model> is not specified.

Directory

empty.vpm should be in the same directory as the layout.

ifosre.exe

Description

Report editor.

Directory

C: \ VPMS \ Wbench \

menubar.ini

Description

Definition of Vframe main menu.

Directory

C: \ VPMS \ Vframe \ menu \

menuburo.ini

Description

Directory

C: \ VPMS \ Vframe \ menu \

menufens.ini

Description

Definition of submenu items for VFrame main menu item Fenster.

Directory

C: \ VPMS \ Vframe \ menu \

menuhelp.ini

Description

Definition of submenu items for VFrame main menu item Help.

Directory

C: \ VPMS \ Vframe \ menu \

menuinfo.ini

Description

Definition of submenu items for VFrame main menu item Infodesk.

Directory

C: \ VPMS \ Vframe \ menu \

menukund.ini

Description

Definition of submenu items for VFrame main menu item Customer.

Directory

C: \ VPMS \ Vframe \ menu \

vds.exe

Description

Designer exec.

Directory

C: \ VPMS \ Designer \

vds_tx32.exe

Description

Designer test exec.

Directory

C: \ VPMS \ Designer \

verwalt.ini

Description

Definition of submenu items for VFrame main menu item Administration.

Directory

C: \ VPMS \ Vframe \ menu \

vframe32.exe

Description

VFrame exec.

Directory

C: \ VPMS \ Vframe \

vorgang.ini

Description

Definition of submenu items for VFrame main menu item Process.

Directory

C: \ VPMS \ Vframe \ menu \

vpms32.exe

Description

Workbench exec.

Directory

C: \ VPMS \ Wbench \

vpmsdl32.dll

Description

Creates a runtime from the .pms file.

Directory

C: \ VPMS \ Wbench \

vpmsdll.dll

Description

16-bit interface to the 16-bit IFOS.

Directory

C: \ Winnt \ System32 \

vpmsin32.dll

Description

32-bit interface to the 32-bit IFOS.

Directory

C: \ Winnt \ System32 \

vpmsinfo.dll

Description

Provides access to the attributes in the designer.

Directory

C: \ Winnt \ System32 \

vpmste32.exe

Description

Workbench test exe.

Directory

C: \ VPMS \ Wbench \

xplconf.ini

Description

Definitions for VFrame.

Directory

C: \ VPMS \ Vframe \ menu \

File types

The following is a list of file types associated with Designer (and VPMS / Windows in general).

- .cat
- .hlp
- .pms

- .vpc
- .vpd
- .vpg
- .vpl
- .vpm

.avi

Description

Multimedia (audio-visual) file type supported by Designer with the Multimedia element.

Typical directory

C: \ VPL_Apps \ (application dir) \

Example

Standard Windows clock.avi file:



Note: Click on the "Stop" button (red with the "X") in the help or browser toolbar to stop playing the .avi file.

.bmp

Description

Windows bit-map picture format. Multimedia(audio-visual) file type supported by Designer with the Multimedia element.

Typical directory

C: \ VPL_Apps \ (application dir) \
000224TTA: doubletext removed.

.cat

Description

Report file. Contains print specifications.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ print \ (application name).cat

.hlp

Description

Help file for consultation in VFrame.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ help \ (help filename).hlp

.jpg

Description

JPEG picture format. Multimedia(audio-visual) file type supported by Designer with the Multimedia element.

Typical directory

C: \ VPL_Apps \ (application dir) \

.pms

Description

Model file. Text file format.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (application name).pms

.tif (tiff)

Description

Tagged-image file format for pictures.

A compressed format with no loss of picture quality.

Multimedia (audio-visual) file type supported by Designer with the Multimedia element.

Typical directory

C: \ VPL_Apps \ (application dir) \

.vpc

Description

Compiled (runtime) layout file. Compiled from .vpl file.

Byte code for distributions of XPL-applications.

As the previous experience has shown the existing VPL-format is rather convenient at design time, but not very suitable at runtime. Most significant disadvantages are low performance during loading an XPL-applications (especially great ones) and ASCII-format that provides possibility to edit VPL-files using usual text editors and does not allow to avoid of distribution of confidential information such as programmer names, comments, etc.

To solve these problems a new binary format (a byte code) of XPL-application should be developed. The following rules are applied to this format:

- The byte code is used only at runtime. At design time the existing VPL-format is used, as well as newly developed format for storing SQL-statements.
- The byte code is generated when 'Save/Save as' function is applied to the application in the Designer Workbench. The byte code differs from VPL-file by its filename extensions.
- The byte code contains all information necessary for an XPL-application at runtime. That means, that it consists not only of VPL-file information, but also of necessary SQL-statements, etc. The byte code of an XPL-application will represent a distribution of the application.
- Load performance of the byte code must be significantly higher than it is for the VPL-format.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (application name).vpc

.vpd

Description

Mapping between layout components and model attributes.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (application name).vpd

.vpg

Description

Contains a group of elements.

Created by selecting a group of elements and then selecting Write to Disk.

Contents can be copied to a layout by:

- Selecting Read group from disk to clipboard.
- Paste the clipboard contents to the layout.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (element group name).vpg

.vpl

Description

Design-time layout . Text file format.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (application name).vpl

Syntax description

VPL-syntax is described in BNF (Bakus Naur Form). The following agreements are used in the VPL-syntax description.

The terminal symbols are written in capitals. The terminal symbols defining values (such as numbers, strings, identifiers, etc) are prefixed with underscore '_'. They are the following.

- **_INT** - integer number (unsigned);
- **_STATUS** - element binary flags (32 binary digits);
- **_VERSION** - version number (version number and sub-number separated with dot '.');
- **_FONTSTYLE** - font style (4 binary digits leading by '@', ex. @0110, strike, underline, italic, bold correspondingly);
- **_COLOR** - color (6 hex digits leading by #, ex. #0A121C , color in RGB-form);
- **_STRING** - string in double-quotes, ex. "Insurance";
- **_IDENTIFIER** - VPL-identifier defining layout object name (led by ':');
- **_VPM_IDENTIFIER** - VPM-identifier passed from product model (led by '\$');
- **_LAYLEAD_IDENTIFIER** - concatenated identifier (VPL-identifier followed by VPM-identifier, ex. :A_ALLG_sFaTelVor\$A_ALLG_sFaTelVor);
- **_VPMLEAD_IDENTIFIER** - concatenated identifier (VPM-identifier followed by VPL-identifier, ex. \$A_PKVV_Nutzart:A_PKVV_Nutzart).

The other terminal symbols define correspondent VPL-keywords.

The changes made in VPL-syntax in version 3.25 (since version 3.1) are outlined in bold characters. Obsolete syntax constructions remaining in the grammar for compatibility with the previous versions are outlined by italic.

```
/* Grammar of application layout description */
/* Common elements */
/* Font description */
font_description
: FONT (' _STRING ' ; _INT ' ; _FONTSTYLE ' ; _COLOR ' )
version_description
: VERSION ( '
_STRING ' ; /* Title */
_STRING ' ; /* Version */
```

```

_STRING ',' /* Description */
_STRING ',' /* Created By */
_STRING ',' /* Creation Date */
_STRING ',' /* Last Updated By */
_STRING ',' /* Last Updated Date */
_STRING ',' /* Last Update Description */ ')
tooltip_description
: TOOLTIP (' _STRING ',' /* Caption */
_COLOR ',' /* Color */
_fontdescription ')
/*-----*/
/* Application layout description -- start symbol */
Application_description
: APPLICATION _IDENTIFIER _VERSION ',' app_property_list ';' workarea_description_list END
app_property_list
: app_property
| app_property_list ',' app_property
app_property
: DESIGNER _VERSION
| USER _STRING
| CAPTION _STRING
| MODEL _STRING _VERSION
| STYLE SIMPLE
| STYLE NOTEBOOK notebook_font_description
| STARTATTRIBUTE _VPM_IDENTIFIER
| STARTVALUE _STRING
| STARTVALUE DYNAMIC
| COMPUTE AUTO
| COMPUTE MANUAL
| FORMAT F640x480 /* Resolution 640x480 */
| FORMAT F800x600 /* Resolution 800x600 */
| FORMAT F1024x768 /* Resolution 1024x768 */
| FORMAT CUSTOM /* Custom resolution */
| ACOLOR _COLOR
| SIZE (' _INT ',' _INT ')
| DEFAULT font_description /* Default application font */
| version_description
notebook_font_description
: /* Empty */
| font_description
/* Workarea description */
workarea_description_list
: workarea_description
| workarea_description_list workarea_description
workarea_description
: WORKAREA _IDENTIFIER ',' workarea_property_list ';' page_description_list ENDAREA ';'
workarea_property_list
: workarea_property
| workarea_property_list ',' workarea_property
workarea_property
: CAPTION _STRING
| STYLE SIMPLE

```

```

| TABCOLOR _COLOR
| HELPID _INT
| INDICATOR (' _VPM_IDENTIFIER ' ; _STRING ' ; _STRING ') /* Data indicator property/attribute,
ON image file, OFF image file */
| STYLE FRAME (' frame_property_list ')
| tooltip_description
| version_description
frame_property_list
: frame_property
| frame_property_list ' ; ' frame_property
frame_property
: WIDTH _INT /* For compatibility with the previous versions only */
| font_description
| WIDTH (' _INT ' ; ' _INT ' ; ' _INT ') /* Middle, minimal and maximal width */
page_description_list
: page_description
| page_description_list page_description
/* Page description */
page_description
: PAGE _IDENTIFIER page_multiple ' ; ' page_property_list ' ; ' page_body ENDPAGE ' ; '
page_body
: /* Empty */
| layout_element_list ' ; '
page_multiple
: /* Empty */
| MULTIPLE
page_property_list
: page_property
| page_property_list ' ; ' page_property
page_property
: CAPTION _STRING
| CAPTION NEW _STRING
| CAPTION DELETE _STRING
| KEYFIELD _IDENTIFIER
| COUNTER _VPM_IDENTIFIER
| INDICATOR (' _VPM_IDENTIFIER ' ; _STRING ' ; _STRING ') /* Data indicator property/attribute,
ON image file, OFF image file */
| LIMIT _VPM_IDENTIFIER
| PARENT WORKAREA /* for frame root page */
| PARENT _IDENTIFIER
| CHECKLIST (' check_list ')
| SEQUENCE STANDARD
| SEQUENCE CUSTOM
| DEFOLD AUTO
| DEFOLD MANUAL
| BGCOLOR _COLOR
| HELPID _INT
| version_description
| tooltip_description
/* Page contens description */
layout_element_list
: /* Empty */

```

```

| layout_element
| layout_element_list ';' layout_element
layout_element
: element_description
| GROUP _IDENTIFIER ';' layout_element_list ';' ENDGROUP
element_description
: EDIT edit_type element_name ';' element_property_list
| CHECKBOX element_name ';' element_property_list
| COMBOBOX element_name ';' element_property_list
| RADIOBUTTON element_name ';' element_property_list
| PUSHBUTTON element_name ';' element_property_list
| RESULT result_type element_name ';' element_property_list
| LABEL element_name ';' element_property_list
| LINE element_name ';' element_property_list
| BORDER element_name ';' element_property_list
| RADIOGROUP element_name ';' element_property_list
| GRID grid_type element_name ';' element_property_list
| PICTURE element_name ';' element_property_list
| DATA element_name data_element_description /* Data element */
| DIAGRAM element_name _INT /* Diagram type */ ';' element_property_list
element_name
: _IDENTIFIER
| _LAYLEAD_IDENTIFIER
| _VPMLEAD_IDENTIFIER
/* EDIT */
edit_type
: TEXT
| NUMBER
| DATE
| MONEY
| MULTILINE
| MASK
/* RESULT */
result_type
: /* Empty - text result */
| NUMBER
| DATE
| MONEY
| MULTILINE
| MASK
grid_type
:
| ACTIVE
/* DATA */
data_element_description
: /* No properties defined */
| ';' element_property_list
element_property_list
: element_property
| element_property_list ';' element_property
element_property
: CAPTION _STRING

```

| POSITION (' _INT ',' _INT ')
| SIZE (' _INT ',' _INT ')
| INPUT (' _INT ',' _INT ') /* Minimal-maximal characters input for text edits */
| font_description
| BGCOLOR _COLOR
| ACOLOR _COLOR
| CURRENCY ON /* Obsolete, now STATUS property is used */
| CURRENCY OFF /* Obsolete, now STATUS property is used */
| LEVEL _INT
| DECIMAL _INT
| HIDE (' page_area_name_list ')
| VISUALIZE (' page_area_name_list ')
| EXECUTE DLL _STRING
| EXECUTE APPLICATION (' _STRING ',' app_session ',' _INT ',' _INT ') /* Pushbuttons - Secondary dialog call: name, clone session flag, dialog window size */
| EXECUTE ACCEPT /* Pushbutton accept dialog */
| EXECUTE CANCEL /* Pushbutton cancel dialog */
| EXECUTE COMPUTE /* Pushbutton compute */
| CHECKLIST (' check_list ')
| COMPUTE ON /* Obsolete, the same as EXECUTE COMPUTE */
| COMPUTE OFF /* Obsolete */
| COMPUTE ONCHANGE /* Obsolete, now STATUS property is used */
| COMPUTE ONEXIT /* Obsolete, now STATUS property is used */
| MULTIPLE ON /* Obsolete, now STATUS property is used */
| MULTIPLE OFF /* Obsolete, now STATUS property is used */
| ALWAYS ACTIVE /* Obsolete, now STATUS property is used */
| STATUS _STATUS
| ALIGNMENT LEFT /* Obsolete, now STATUS property is used */
| ALIGNMENT RIGHT /* Obsolete, now STATUS property is used */
| STYLE SIMPLE
| STYLE SHADOW
| CONVERT _VPM_IDENTIFIER
| CONVERT ON _VPM_IDENTIFIER
| CONVERT OFF _VPM_IDENTIFIER
| FILE _STRING
| HELPID _INT
| COLUMNS _INT
| KEYCOLUMN _INT
| ITERATION iteration_type
| STEP _INT
| HEADER font_description
| column_type (' column_property_list ')
| CHECKVISIBLE _VPM_IDENTIFIER /* check visibility of result fields */
| CAPTION LEFT _STRING
| CAPTION RIGHT _STRING
| CAPTION UP _STRING
| CAPTION DOWN _STRING
| COUNTER _VPM_IDENTIFIER
| CAPTION NEW _STRING
| CAPTION DELETE _STRING
| PICTURE (' _STRING ',' _STRING ',' _STRING ',' _STRING ')
| version_description


```
page_area_name_list
: page_area_name
| page_area_name_list ',' page_area_name
page_area_name
: _IDENTIFIER area_mark
area_mark
: /* Empty */
| (' ') /* Here area name in the list */
check_list
: check_name
| check_list ',' check_name
;
check_name
: _IDENTIFIER /* Obsolete, for compatibility with previous VPL versions only */
| _VPM_IDENTIFIER
app_session
: ON
| OFF
column_type
: INPUT column_cell_type COLUMN
| OUTPUT column_cell_type COLUMN
column_cell_type
: /* Empty */
| NUMBER
| DATE
| MONEY
| COMBOBOX
iteration_type
: ATTRIBUTE _VPM_IDENTIFIER
| PROPERTY _VPM_IDENTIFIER
| COUNTER_INT
column_property_list
: column_property
| column_property_list ',' column_property
column_property
: CAPTION_STRING
| WIDTH_INT
| SOURCE _VPM_IDENTIFIER
| NEXT_INT
| font_description
| BGCOLOR_COLOR
| DECIMAL_INT
| ALIGNMENT RIGHT
| CURRENCY OFF
| STATUS_STATUS
| CHECKVISIBLE _VPM_IDENTIFIER
| CHECKLIST LP check_list RP
| MASK_STRING
| INPUT LP_INT CM_INT RP
/* The End */
```

.vpm

Description

Compiled (runtime) model file. Compiled from .pms.
Setting default path for .vpm files.

Typical directory / filename

C: \ VPL_Apps \ (application dir) \ (application name).vpm

.wav

Description

Windows audio-visual format. Multimedia(audio-visual) file type supported by Designer with the Multimedia element.

Typical directory

C: \ VPL_Apps \ (application dir) \

Example

Assume the following layout Media element.

In the runtime layout chimes.wav would play continuously whenever the page is selected.